{ LUIS JOSÉ SÁNCHEZ }

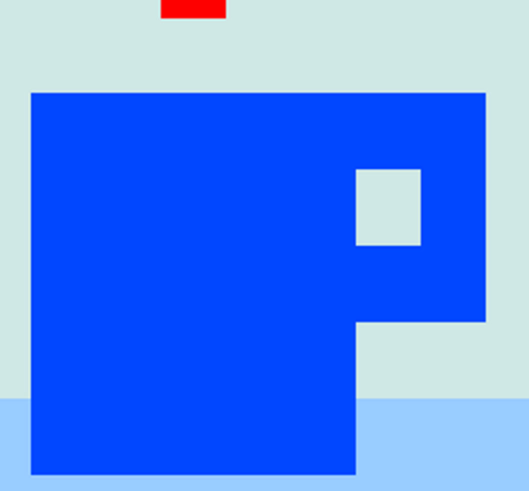


APRENDE JAVA CON EJERCICIOS



EDICIÓN 2019

350 EJERCICIOS RESUELTOS



Aprende Java con Ejercicios

Luis José Sánchez González

Este libro está a la venta en http://leanpub.com/aprendejava

Esta versión se publicó en 2019-09-22

ISBN 978-84-608-2372-8



Este es un libro de Leanpub. Leanpub anima a los autores y publicadoras con el proceso de publicación. Lean Publishing es el acto de publicar un libro en progreso usando herramientas sencillas y muchas iteraciones para obtener feedback del lector hasta conseguir tener el libro adecuado.

Índice general

Sobre este libro

"Aprende Java con Ejercicios" es un manual práctico para aprender a programar en Java desde cero.

No es necesario tener conocimientos previos de programación. La dificultad del libro es gradual, empieza con conceptos muy básicos y ejercicios muy sencillos y va aumentando en complejidad y dificultad a medida que avanzan los capítulos.

La práctica hace al maestro. Como de verdad se aprende a programar es programando desde el principio y esa es la filosofía que sigue este libro. En cada capítulo se explican una serie de conceptos de programación y se ilustran con ejemplos. Al final de cada uno de los capítulos se plantean ejercicios de diferente dificultad.

Este libro contiene 350 ejercicios. Tanto las soluciones a los ejercicios como los ejemplos están disponibles en GitHub: https://github.com/LuisJoseSanchez/aprende-java-con-ejercicios y tanto los unos como los otros siguen los estándares de programación de Google para el código fuente escrito en el lenguaje de programación Java[1]. Las soluciones a los ejercicios también están disponibles en el Apéndice E y se actualizan con cada edición del libro en base al repositorio de GitHub antes mencionado.

"Aprende Java con Ejercicios" es un libro hecho casi a medida de la asignatura *"Programación"* que forma parte del currículo del primer curso de los ciclos formativos DAW (Desarrollo de Aplicaciones Web) y DAM (Desarrollo de Aplicaciones Multiplataforma) pero igualmente puede ser utilizado por estudiantes de Ingeniería Informática, Ingeniería de Telecomunicaciones o Ciencias Matemáticas en la asignatura de *"Programación"* de los primeros cursos.

---

[1] https://google.github.io/styleguide/javaguide.html

Sobre el autor

Luis José Sánchez González es Ingeniero Técnico en Informática de Gestión por la Universidad de Málaga (España) y profesor de la Junta de Andalucía con más de 20 años de experiencia. En su trabajo de profesor de Informática combina sus dos pasiones: la enseñanza y la programación.

En el momento de publicar este libro, es profesor del **I.E.S. Campanillas (Málaga)** e imparte clases en el **Ciclo Superior de Desarrollo de Aplicaciones Web** y en el **Ciclo Superior de Desarrollo de Aplicaciones Multiplataforma** en la sede que tiene el I.E.S. Campanillas[2] en el Parque Tecnológico de Andalucía[3].

Puedes ponerte en contacto con el autor mediante la dirección de correo electrónico luisjoseprofe@gmail.com o mediante LinkedIn (https://www.linkedin.com/in/luisjosesanchez/).

---

[2] http://fp.iescampanillas.com/

[3] http://www.pta.es

Centros en los que se usa este libro

A continuación se listan los centros donde se utiliza este libro, bien como libro de texto, como libro auxiliar o como libro de apoyo del profesor.

Si eres alumno o profesor y utilizas este libro para aprender o enseñar Java, pídele al autor que añada en esta sección el nombre de tu centro educativo o empresa, escribiendo un correo a la dirección luisjoseprofe@gmail.com indicando la localidad y el país del centro.

Almería (España)

- IES Murgi (El Ejido)

Córdoba (España)

- IES Álvarez Cubero (Priego de Córdoba)
- IES Gran Capitán (Córdoba)

Huelva (España)

- IES San José (Cortegana)

Jaén (España)

- IES Castillo de la Yedra (Cazorla)
- IES Fernando III (Martos)
- IES Jándula (Andújar)
- IES Las Fuentezuelas (Jaén)

Málaga (España)

- IES Campanillas
- IES Politécnico Jesús Marín
- IES Portada Alta

Santa Fe (Argentina)

- Escuela de Educación Técnico Profesional N° 647 Pedro L. Funes

Sevilla (España)

- IES El Majuelo (Ginés)
- IES Hermanos Machado (Dos Hermanas)
- IES Jacarandá (Brenes)
- IES Martínez Montañés (Sevilla)
- IES Polígono Sur (Sevilla)
- IES Punta del Verde (Sevilla)
- IES Ramón del Valle Inclán (Sevilla)
- IES Ruiz Gijón de Utrera (Sevilla)
- IES Virgen de Valme (Dos Hermanas)

Tegucigalpa (Honduras)

- Universidad José Cecilio del Valle

El libro original

Este libro está a la venta en https://leanpub.com/aprendejava. Una vez comprado el libro, cuando se publica una nueva versión, el lector recibe una notificación para poder descargarla sin coste adicional y sin límite de tiempo.

Si se ha descargado o copiado este libro de otra fuente, puede que no sea la última versión y, por tanto, puede que contenga errores o le falte contenido. La descarga del libro desde la página oficial indicada arriba garantiza disfrutar siempre de la versión más actualizada.

Los profesores pueden solicitar una copia gratuita y cupones de descuento para sus alumnos (para grupos a partir de 10 alumnos) contactando con el autor.

Introducción

Java es un lenguaje de programación; es de hecho, desde hace más de 10 años, el lenguaje de programación más utilizado en el mundo según el índice PYPL[4](PopularitY of Programming Language index).

Los ordenadores no entienden - por ahora - el español, ni el inglés, ni ningún otro idioma natural. De una forma muy simplificada podríamos decir que un lenguaje de programación es un idioma que entiende el ordenador[5]. Cualquier aplicación - procesador de textos, navegador, programa de retoque fotográfico, etc. - está compuesta de una serie de instrucciones convenientemente empaquetadas en ficheros que le dicen al ordenador de una manera muy precisa qué tiene que hacer en cada momento.

Java es un lenguaje de programación estructurado y, como tal, hace uso de variables, sentencias condicionales, bucles, funciones… Java es también un lenguaje de programación orientado a objetos y, por consiguiente, permite definir clases con sus métodos correspondientes y crear instancias de esas clases. Java no es un lenguaje de marcas como HTML o XML aunque puede interactuar muy bien con ellos[6].

Las aplicaciones Java se suelen compilar a *bytecode*, que es independiente del sistema operativo, no a binario que sí depende del sistema operativo. De esta forma, el *bytecode* generado al compilar un programa escrito en Java debería funcionar en cualquier sistema operativo que tenga instalada una máquina virtual de java (JVM).

Cualquier editor simple como **Nano** o **GEdit** es suficiente para escribir código en Java aunque se recomiendan IDEs[7] como **NetBeans** o **Eclipse** ya que tienen algunas características que facilitan mucho la programación como el chequeo de errores mientras se escribe, el autocompletado de nombres de variables y funciones y mucho más.

---

[4] http://pypl.github.io/PYPL.html

[5] En realidad un ordenador no entiende directamente los lenguajes de programación como Java, C, Ruby, etc. Mediante una herramienta que se llama compilador o traductor según el caso, lo que hay escrito en cualquiera de estos lenguajes se convierte en código máquina - secuencias de unos y ceros - que es el único lenguaje que de verdad entiende el ordenador.

[6] El lector deberá tener unos conocimientos básicos de HTML para entender bien y sacar provecho del capítulo relativo a JSP de este mismo libro. Una web excelente para aprender HTML es http://w3schools.com

[7] IDE: Integrated Development Enviroment (Entorno de Desarrollo Integrado)

Instalación y configuración del entorno de programación Java

Todos los ejemplos que contiene este libro así como las soluciones a los ejercicios se han escrito sobre **Linux**, en concreto sobre **Ubuntu** (http://www.ubuntu.com/). No obstante, todo el código debería compilar y funcionar sin ningún problema en cualquier otra plataforma[8].

Es muy recomendable trabajar con **máquinas virtuales**. Esto tiene la ventaja de no "ensuciar" el sistema operativo del ordenador instalando programas y modificando configuraciones que pueden incluso dejarlo inutilizado. Una máquina virtual es básicamente un ordenador simulado dentro de otro mediante un software de virtualización. Se pueden crear tantas máquinas virtuales como se quiera. Si, haciendo pruebas, una de esas máquinas se rompe; no hay ningún problema en borrarla y crear otra nueva.

Sea cual sea el sistema operativo que tengas instalado en tu equipo, antes de seguir con el manual, te recomendaría instalar un software de virtualización, por ejemplo **VirtualBox**[9]. Una vez instalaldo **VirtualBox** y el correspondiente **VirtualBox Extension Pack**, descarga la imagen de la última versión de **Ubuntu** y crea una máquina virtual con **Ubuntu** para instalar y configurar los programas que se especificarán más adelante. No tengas miedo de hacer pruebas en la máquina virtual ya que, como hemos dicho, si se rompe, se puede borrar y crear otra muy fácilmente.

La virtualización se sale del ámbito de este manual. No obstante, tu profesor de Programación te podrá orientar sobre cómo instalarlas y configurarlas. En la página oficial de **VirtualBox** encontrarás un manual detallado sobre el tema[10]: .

Ya sea sobre el sistema real o sobre una máquina virtual (preferiblemente esta última), el software necesario para seguir este manual es el siguiente[11]:

OpenJDK

El **Open Java Development Kit** (OpenJDK) contiene una serie de componentes que permiten desarrollar y ejecutar aplicaciones escritas en Java. Incluye la máquina

[8] En casos muy puntuales, hay pequeñas diferencias en el código dependiendo de la plataforma - Linux, Mac OS X o Windows. Llegado el momento, se indicará convenientemente.

[9] Descarga de **VirtualBox**: https://www.virtualbox.org/. Descarga de **VirtualBox Extension Pack**: https://www.virtualbox.org/wiki/Downloads.

[10] Manual de usuario de **VirtualBox**: http://download.virtualbox.org/virtualbox/UserManual.pdf

[11] Todo los programas recomendados en este manual son libres y se pueden descargar de forma legal y gratuita.

virtual de Java (JVM) que permite ejecutar *bytecode*. Como mencionamos en el apartado Introducción, el *bytecode* es el ejecutable en Java.

El **OpenJDK** también incluye el compilador `javac` que permite compilar el código fuente para generar el *bytecode*.

En **Ubuntu**, basta ejecutar una línea en la ventana de terminal para instalar el **OpenJDK**:

```
sudo apt-get install openjdk-8-jdk
```

Para los sitemas operativos **Mac OS X** y **Windows**, se encuentran disponibles para su descarga los archivos de instalación del **OpenJDK** en la página Unofficial OpenJDK installers for Windows, Linux and Mac OS X[12].

Otra opción consiste en utilizar el **JDK oficial de Oracle**, que se puede descargar en el apartado Java SE Development Kit 8 - Downloads[13] de la web de esta misma empresa.

No vamos a explicar las diferencias entre el **OpenJDK** y el **JDK** ya que se escapa al ámbito de este libro. A efectos prácticos, todos los ejemplos contenidos en este manual funcionan tanto con en el **OpenJDK** como con el **JDK** y los ejercicios se pueden realizar sin problema sea cual sea el kit de desarrollo instalado.

En versiones antiguas de Windows, después de instalar el **JDK** es necesario añadir una variable de entorno[14].

Editor de texto

Para seguir este manual recomendamos utilizar al principio un editor de textos sencillo para probar los primeros programas y realizar los primeros ejercicios, compilando en línea de comandos. De esta manera, el lector se va familiarizando con el proceso de edición de código, compilación y ejecución y va entendiendo lo que sucede en cada paso. En **Ubuntu** viene instalado por defecto el editor **GEdit**, el sistema operativo **Mac OS X** viene con **TextEdit** y en **Windows** podemos usar el programa **Bloc de notas**.

Geany

Una vez que el lector ha superado la barrera de escribir sus primeros programas, recomendamos utilizar el entorno de programación **Geany** (por ejemplo a partir del

[12] https://github.com/alexkasko/openjdk-unofficial-builds
[13] http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
[14] JDK Installation for Microsoft Windows

capítulo 2). Se trata de un IDE muy ligero que permite realizar aplicaciones sencillas con rapidez. Mediante **Geany** se puede escribir el código, chequear si contiene errores, compilar y ejecutar el *bytecode* generado. Para programas pequeños - por ej. calcular la media de varios números - es recomendable utilizar **Geany** en lugar de un IDE más complejo como **Netbeans**. Mientras con **Geany** da tiempo a escribir el código, compilar y ejecutar, **Netbeans** todavía estaría arrancando.

Instalación de **Geany** (y los *plugins* adicionales) en **Ubuntu**:

```
sudo apt-get install geany
sudo apt-get install geany-plugins
```

En la sección de descargas de la página oficial de Geany[15] se encuentran los ficheros de instalación de este programa tanto para **Mac OS X** como para **Windows**.

Una vez instalado **Geany** es conveniente realizar algunos ajustes en la configuración. Presiona **Control + Alt + P** para abrir la ventana de preferencias. Haz clic en la pestaña **Editor** y luego en la pestaña **Sangría**. Establece las opciones tal y como se indican a continuación.

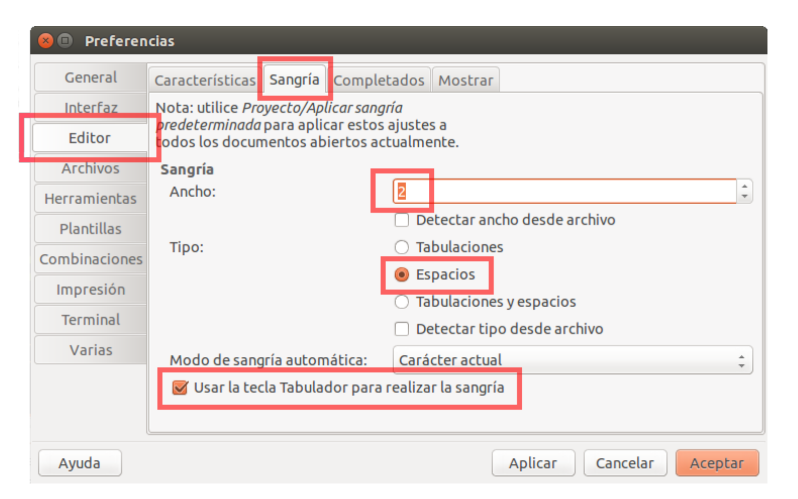


Figura 0.1: Configuración de Geany (sangría)

También es recomendable marcar las casillas **Mostrar guías de sangría** y **Mostrar números de línea**. Estas opciones se activan en la pestaña **Mostrar**.

[15] http://www.geany.org/Download/Releases

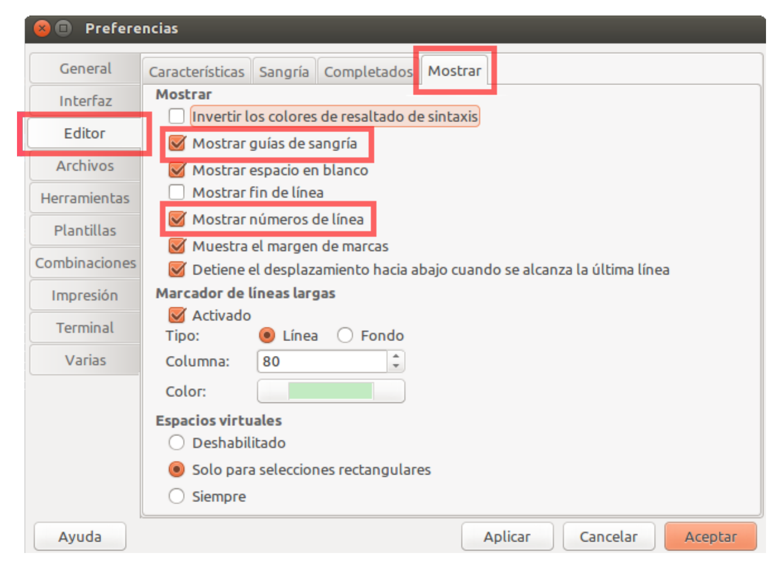


Figura 0.2: Configuración de Geany (guías de sangría y números de línea)

Visual Studio Code

Se trata de un IDE muy ligero y, al mismo tiempo, muy potente que viene por defecto con soporte para JavaScript, TypeScript y Node.js. Se le pueden añadir muchas extensiones para poder trabajar con otros lenguajes de programación como Java, PHP, Python, etc.

Visual Studio Code se puede descargar de forma gratuita para Linux, Windows y Mac OS X desde https://code.visualstudio.com/

Una vez instalado VS Code. Hay que añadir el soporte para Java (figura 0.3).

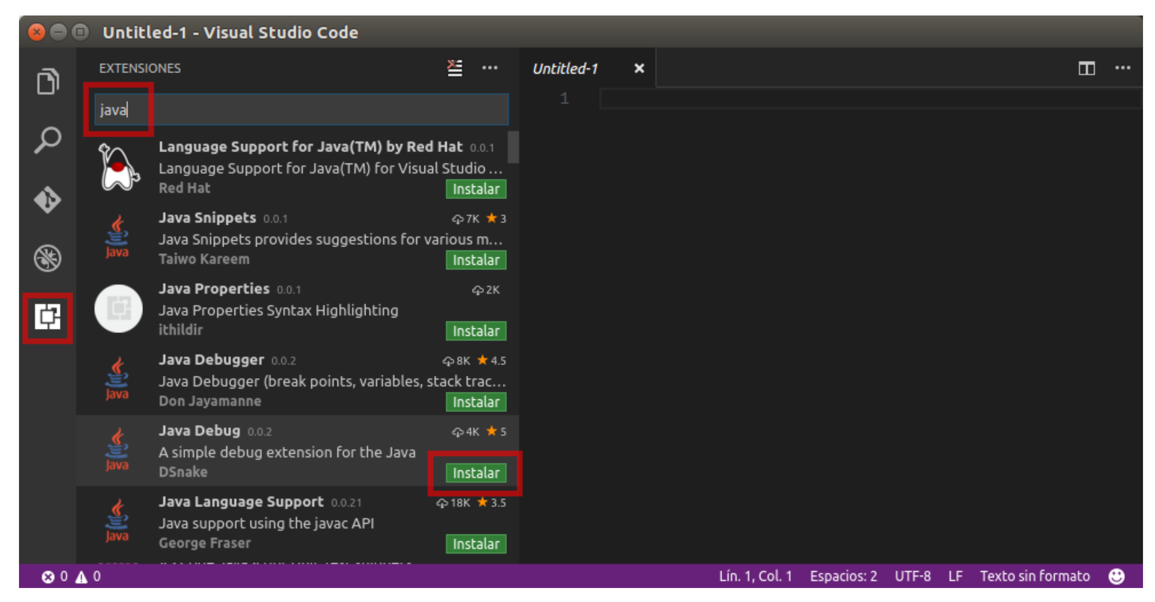


Figura 0.3: Instalación de la extensión Java Debug en VS Code

Haz clic en el icono de extensiones (figura 0.3) para mostrar todas extensiones disponibles. Escribe `java` en el cuadro de búsqueda para filtrar las extensiones relacionadas con Java. Localiza una extensión llamada **Java Debug** y haz clic en el botón **Instalar**.

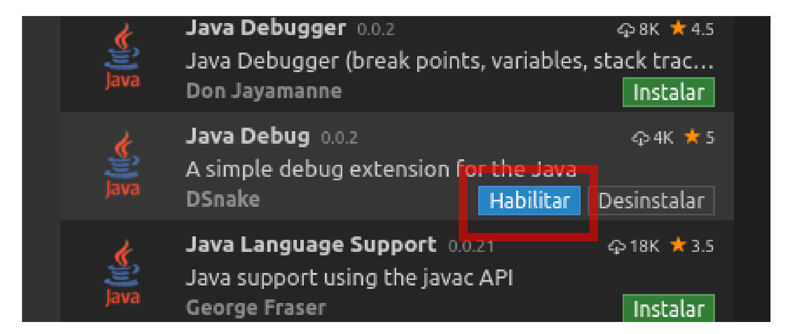


Figura 0.4: Habilitación de la extensión Java Debug en VS Code

Cuando la extensión seleccionada se descarga y se instala. Se le pregunta al usuario si se quiere habilitar. Haz clic en el botón **Habilitar** (figura 0.4) para habilitar la extensión **Java Debug**.

Netbeans

Cuando las aplicaciones a desarrollar son más complejas y, sobre todo, cuando estas aplicaciones están implementadas en varios ficheros, se recomienda utilizar **Netbeans** de la empresa **Oracle**. En el capítulo 9, el lector ya podría sacarle mucho partido al uso de este IDE. Se trata de un entorno integrado muy potente y muy usado a nivel profesional que incluye el autocompletado de sintaxis y permite una

depuración avanzada paso a paso. **Netbeans** se puede descargar de forma gratuita y para cualquier plataforma desde https://netbeans.org/downloads/.

Después de la instalación de **Netbeans**, procedemos a configurar el editor. Para ello seleccionamos **Herramientas** → **Opciones** → **Editor**. A continuación seleccionamos la pestaña **Formato** y en el menú desplegable correspondiente al lenguaje de programación seleccionamos **Java**. La configuración debería quedar como se indica en la figura 0.5).

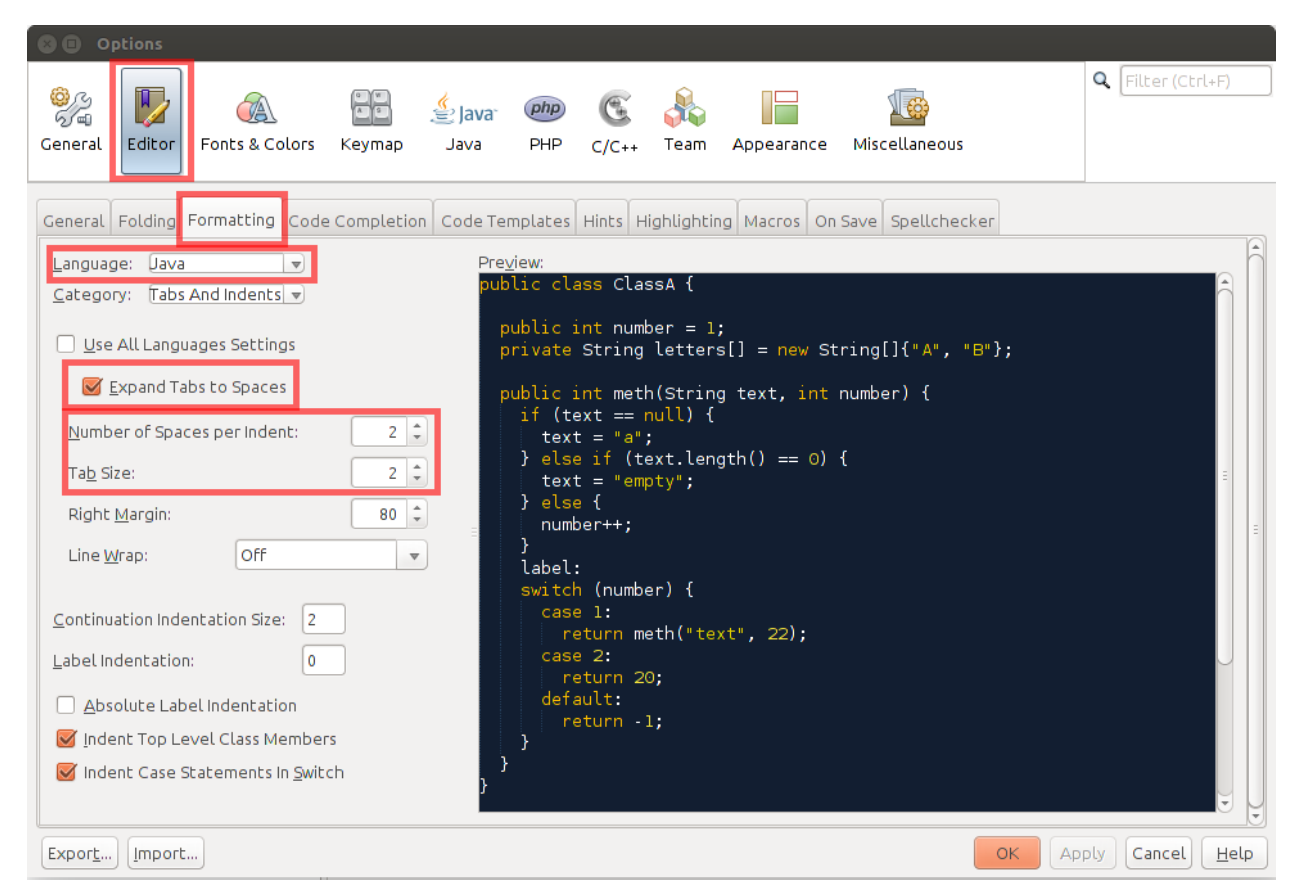


Figura 0.5: Configuración de Netbeans



Es muy importante que los editores/IDEs que se utilicen para esribir código en Java se configuren adecuadamente. Hemos visto cómo configurar **Geany**, **Visual Studio Code** y **Netbeans**, pero si el lector decide utilizar otras herramientas deberá configurarlas igualmente. En definitiva, la configuración/preferencias de las herramientas que se utilicen deben establecerse de tal forma que se cumplan estos requisitos:

- La tecla TAB debe insertar espacios, no debe insertar el carácter de tabulación.
- La indentación debe ser de 2 caracteres (por defecto suele estar a 4).
- La condificación de caracteres debe ser UTF-8.

1. ¡Hola mundo! - Salida de datos por pantalla

1.1 ¡Hola mundo! - Mi primer programa

El primer programa que aprende a hacer cualquier aspirante a programador es un **Hola mundo**[1]. Es seguramente el programa más sencillo que se puede escribir. Se trata de un programa que muestra el mensaje “Hola mundo” por pantalla. Abre el programa **GEdit** que, como hemos comentado en la sección Instalación y configuración del entorno de programación Java, viene instalado por defecto en Ubuntu y teclea el siguiente código:

```
/**
 * Muestra por pantalla la frase "¡Hola mundo!"
 *
 * @author Luis J. Sánchez
 */
public class HolaMundo { // Clase principal
  public static void main(String[] args) {
    System.out.println("¡Hola mundo!");
  }
}
```

Verás que, en principio, el texto del código no aparece en colores. El coloreado de sintaxis es muy útil como comprobarás más adelante, sobre todo para detectar errores. Graba el programa como `HolaMundo.java`. Al grabar con la extensión `.java`, ahora el programa **GEdit** sí sabe que se trata de un programa escrito en Java y reconoce la sintaxis de este lenguaje de forma que puede colorear los distintos elementos que aparecen en el código.

[1] En la página http://helloworldcollection.de/ se muestra el programa ***Hola mundo*** escrito en más de 400 lenguajes de programación.

HolaMundo.java (~/Dropbox/ProyectosGit/aprende-java-con-ej.../01_Hola_mundo_Salida_de_datos_por_pantalla) - gedit

Abrir Guardar Deshacer

HolaMundo.java

```java
/**
 * Muestra por pantalla la frase "¡Hola mundo!"
 *
 * @author Luis J. Sánchez
 */
public class HolaMundo { // Clase principal
  public static void main(String[] args) {
    System.out.println("¡Hola mundo!");
  }
}
```

Jav

Figura 1.1: Programa `Hola mundo` escrito en el editor GEdit

Fíjate que el nombre que le damos al fichero es exactamente igual que la clase principal seguido de la extensión `.java`. Abre una ventana de terminal, entra en el directorio donde se encuentra el fichero y teclea lo siguiente:

```
javac HolaMundo.java
```

Este comando crea `HolaMundo.class` que es el *bytecode*. Para ejecutar el programa, teclea:

```
java HolaMundo
```

¡Enhorabuena! Acabas de realizar tu primer programa en Java.

En la siguiente captura de pantalla puedes ver la ventana de terminal y los comandos que acabamos de describir.

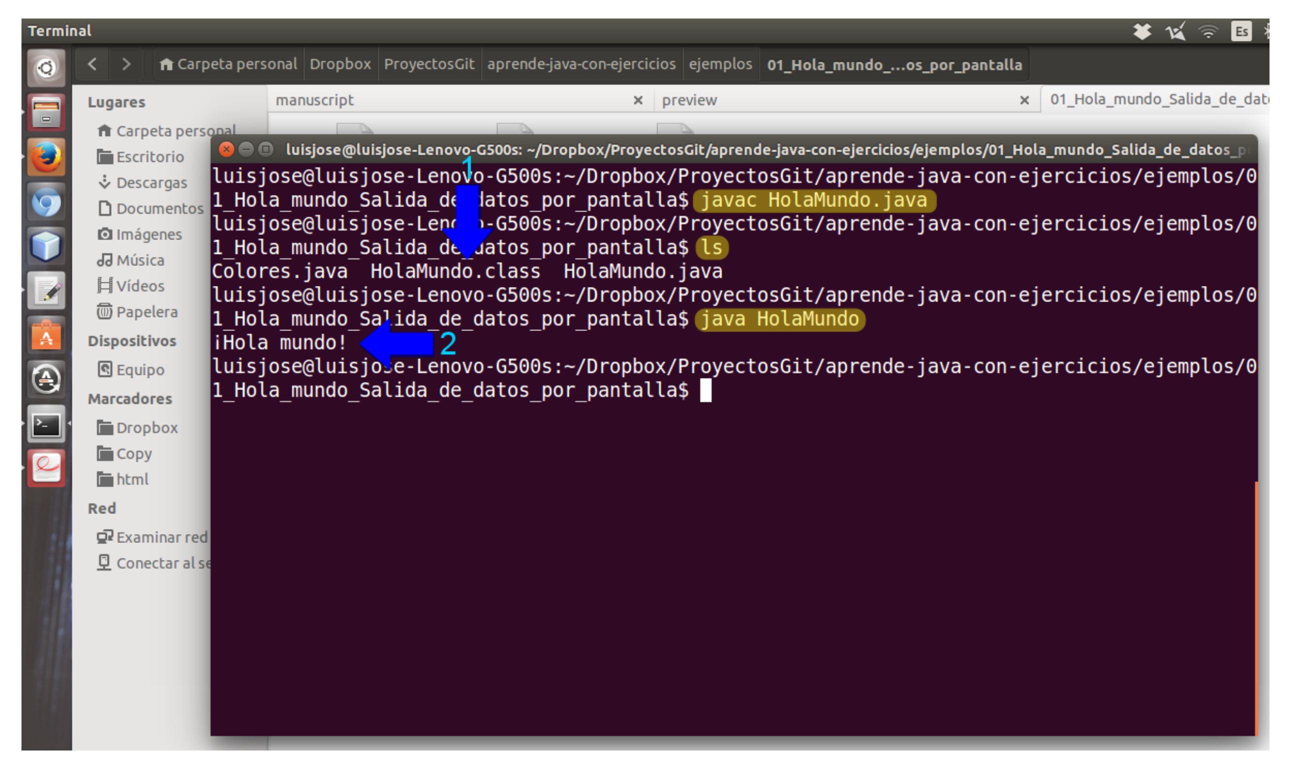


Figura 1.2: `HolaMundo.class` es el *bytecode* generado (1). El resultado del programa es una línea de texto escrita en pantalla (2).

La instrucción que hemos utilizado para mostrar una frase por pantalla es `System.out.println()`, colocando la frase entre paréntesis. También se puede volcar texto por pantalla mediante `System.out.print()`. La única diferencia radica en que esta última no añade un salto de línea al final, pruébalo en `HolaMundo.java` y compruébalo por ti mismo. Si lo que quieres mostrar es una palabra o una frase, como en este ejemplo, es importante que el texto esté entrecomillado.



- El programa propiamente dicho está dentro de lo que llamamos "clase principal".
- El fichero debe tener el mismo nombre que la clase principal seguido por la extensión `.java`
- Los comentarios de varias líneas se colocan entre `/*` y `*/`
- Los comentarios de línea se indican mediante `//`
- Para mostrar información por pantalla se utilizan `System.out.println()` y `System.out.print()`.

1.2 Coloreado de texto

Mostrar siempre texto blanco sobre fondo negro puede resultar muy aburrido. El texto que se muestra por pantalla se puede colorear; para ello es necesario insertar unas secuencias de caracteres - que indican el color con el que se quiere escribir - justo antes del propio texto[2].

Prueba el siguiente programa. Observa que delante de cada palabra hay una secuencia de caracteres. Esa secuencia provoca un cambio de color.

```
/**
 * Coloreado de texto
 *
 * Muestra varias palabras en el color que corresponde.
 *
 * @author Luis José Sánchez
 */
public class Colores {
  public static void main(String[] args) {
    System.out.print("\033[33m mandarina");
    System.out.print("\033[32m hierba");
    System.out.print("\033[31m tomate");
    System.out.print("\033[37m sábanas");
    System.out.print("\033[36m cielo");
    System.out.print("\033[35m nazareno");
    System.out.print("\033[34m mar");
  }
}
```

A continuación se muestra el resultado de la ejecución del programa.

[2] El coloreado de texto usando este método puede que no funcione en terminales de otros sistemas operativos que no son Linux.

```
mandarina hierba tomate sábanas cielo nazareno mar

------------------
(program exited with code: 0)
Press return to continue
```

Figura 1.3: Coloreado de texto.

Se ha escrito un espacio entre cada secuencia que establece el color y cada palabra por claridad, pero hay que tener en cuenta que esos espacios se "pintan", es decir, se colocan huecos. Se podría prescindir del espacio sin ningún problema, por ejemplo escribiendo `System.out.print("\033[31mtomate")` en lugar de `System.out.print("\033[31m tomate")`.

Veamos la gama de colores disponibles.

```java
/**
 * Tabla con los colores disponibles
 *
 * Muestra varias palabras en el color que corresponde.
 *
 * @author Luis José Sánchez
 */
public class TablaDeColores {
  public static void main(String[] args) {
    System.out.println("┌─────────┬────────────┬─────────┬─────────────────┐");
    System.out.println("│ Código │    Color    │ Código │      Color       │");
    System.out.println("├─────────┼────────────┼─────────┼─────────────────┤");
    System.out.print("│   30   │\033[30m negro \033[39;49m    │");
    System.out.println("   90   │\033[90m negro claro \033[39;49m   │");
    System.out.print("│   31   │\033[31m rojo \033[39;49m     │");
    System.out.println("   91   │\033[91m rojo claro \033[39;49m    │");
    System.out.print("│   32   │\033[32m verde \033[39;49m    │");
    System.out.println("   92   │\033[92m verde claro \033[39;49m   │");
    System.out.print("│   33   │\033[33m amarillo \033[39;49m │");
    System.out.println("   93   │\033[93m amarillo claro \033[39;49m│");
    System.out.print("│   34   │\033[34m azul \033[39;49m     │");
    System.out.println("   94   │\033[94m azul claro \033[39;49m    │");
    System.out.print("│   35   │\033[35m morado \033[39;49m   │");
    System.out.println("   95   │\033[95m morado claro \033[39;49m  │");
    System.out.print("│   36   │\033[36m cian \033[39;49m     │");
    System.out.println("   96   │\033[96m cian claro \033[39;49m    │");
    System.out.print("│   37   │\033[37m blanco \033[39;49m   │");
```

```
    System.out.println("   97   │\033[97m blanco claro \033[39;49m  │");
    System.out.println("└────────┴───────────┴────────┴────────────────┘");
  }
}
```

El programa muestra una tabla por consola con los códigos de color disponibles. Usando la plantilla `\033[XXm`, basta cambiar `XX` por código correspondiente al color deseado. Estos códigos están establecidos en el estándar ANSI[3].

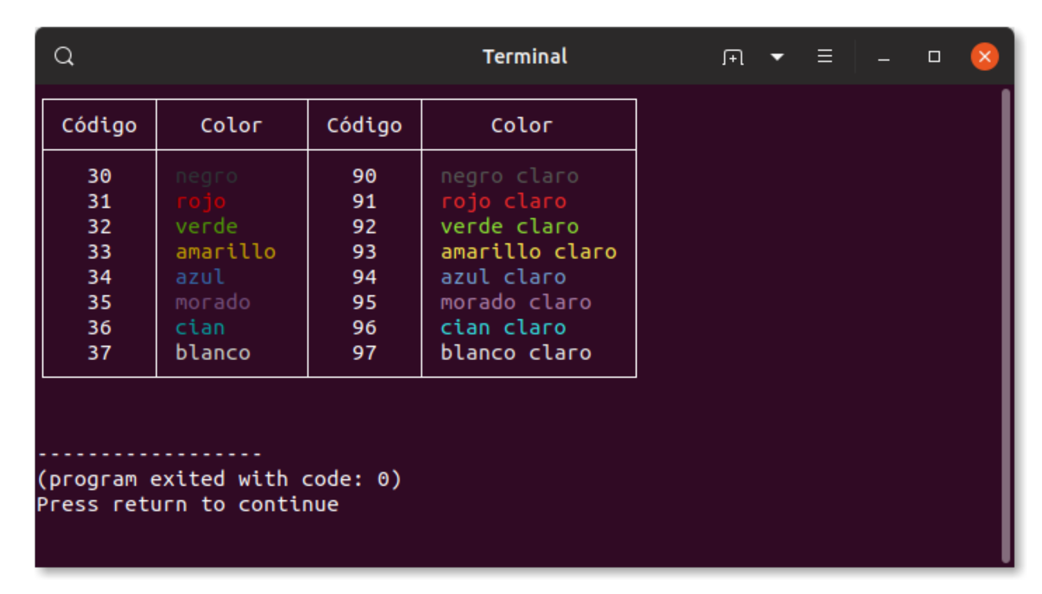


Figura 1.3: Coloreado de texto.

1.3 Sangría

Como puedes ver en todos los ejemplos que hemos mostrado, algunas líneas están ligeramente desplazadas hacia la derecha, es lo que se llama **sangría** o **indentación**. En programación es muy importante sangrar (indentar) bien porque da una idea de qué partes del código son las que contienen a otras. En el último ejemplo tenemos un programa que tiene unos comentarios al principio y luego la clase principal marcada por la línea `public class TablaDeColores {`. Dentro de la clase `TablaDeColores` está el `main` o bloque del programa principal que tiene sangría por estar dentro de la definición de la clase `Colores`. A su vez, dentro del `main` hay una serie de líneas de código que igualmente tienen sangría.

En este libro, tanto el código de los ejemplos como el de las soluciones a los ejercicios siguen el estándar de Google para el código fuente escrito en el lenguaje de programación Java[4], por tanto, cada vez que se aplique la sangría será exactamente

[3] https://en.wikipedia.org/wiki/ANSI_escape_code
[4] https://google.github.io/styleguide/javaguide.html

de **dos espacios**.

1.4 Caracteres especiales

Mediante `System.out.print()` se pueden mostrar palabras o frases, esto es, secuencias de letras, espacios y signos de puntuación como hemos visto. Esta instrucción, además, permite mostrar caracteres especiales - emoticonos, figuras de ajedrez, fichas de dominó, etc. En el Apéndice C. Caracteres Especiales tienes una selección con muchos de estos caracteres clasificados en diferentes categorías.

Los caracteres especiales se pueden copiar y pegar en el código fuente como se muestra en el siguiente ejemplo[5].

```
Archivo  Editar  Ver  Ir  Ayuda
CaracteresEspeciales.java ×

/**
 * Muestra caracteres especiales
 *
 * @author Luis J. Sánchez
 */
public class CaracteresEspeciales {
  public static void main(String[] args) {
    System.out.print("Cuando programo ⌨ estoy");
    System.out.print(" muy 😆 y me siento como un ♔.");
  }
}

SALIDA                                   Java Debug
Cuando programo ⌨ estoy muy 😆 y me siento como un ♔.

⊗ 0 ⚠ 0       Lín. 1, Col. 4  Espacios: 2  UTF-8  LF  Java
```

Figura 1.4: Caracteres especiales.

En lugar de copiar y pegar, se puede incluir un carácter especial en una cadena de caracteres si se sabe su código en el estándar **Unicode**[6]. Basta con escribir `\u` seguido del código. Por ejemplo, el símbolo de la corchea tiene el código unicode `266A` y el de doble corchea `266A`.

[5] El código de ejemplo funciona en Linux, podría no funcionar en otras plataformas.

[6] Lista de caracteres Unicode: https://en.wikipedia.org/wiki/List_of_Unicode_characters

```
/**
 * Muestra caracteres especiales
 *
 * @author Luis J. Sánchez
 */
public class CaracteresEspeciales02 {
  public static void main(String[] args) {
    System.out.println("Una corchea: \u266A");
    System.out.print("Dos corcheas: \u266B");
  }
}
```

```
Una corchea: ♪
Dos corcheas: ♫
```

Figura 1.5: Caracteres especiales.

1.5 Salida formateada

Java dispone de la instrucción `System.out.printf()` análoga al `printf()` del lenguaje C. Permite formatear la salida que se pretende mostrar por pantalla.

El ejemplo que se muestra a continuación:

```java
/**
 * Salida formateada
 *
 * @author Luis J. Sánchez
 */
public class SalidaFormateada01 {
  public static void main(String[] args) {
    System.out.printf("El número %d no tiene decimales.\n", 21);
    System.out.printf("El número %f sale con decimales.\n",  21.0);
    System.out.printf("El %.3f sale exactamente con 3 decimales.\n", 21.0);
  }
}
```

muestra la siguiente salida:

```
El número 21 no tiene decimales.
El número 21,000000 sale con decimales.
El 21,000 sale exactamente con 3 decimales.
```

Veamos en detalle una de las líneas.

```
System.out.printf("El número %d no tiene decimales.\n", 21);
```

Observa que a la instrucción `System.out.printf()` se le pasan dos elementos; por un lado, una plantilla entrecomillada - `"El número %d no tiene decimales.\n"` - y por otro lado el/los dato/s separado/s por coma. En este caso únicamente se pasa un dato - el número 21 - pero podrían ser varios números o incluso números y palabras. Lo que sucede en esta línea es que el dato - el número 21 - se inserta dentro de la plantilla, justo en la casilla `%d`. Debe haber tantas casillas con el símbolo `%` como datos separados por coma. Los datos se insertan siempre por orden en las casillas que le corresponden. En una casilla `%d` solo se aceptan números enteros (sin decimales).

El carácter `'\n'` es un salto de línea.

Veamos ahora otra de las líneas del programa.

```
System.out.printf("El %.3f sale exactamente con 3 decimales.\n", 21.0);
```

La casilla `%f` se debe rellenar con un número que contiene decimales. Si se escribe `%f` tal cual, se mostrarán varios decimales, aunque no sabemos en principio cuántos. Mediante `%.3f` se especifica que se deben mostrar exactamente 3 decimales.

La casilla `%s` dentro de una plantilla que se le pasa a un `System.out.printf()` se debe rellenar con una cadena de caracteres, es decir, con una secuencia de letras, espacios y signos de puntuación.

Veamos un ejemplo en el que se mezcla la salida formateada de cadenas de caracteres, números enteros y números con decimales.

```
/**
 * Salida formateada
 *
 * @author Luis J. Sánchez
 */
public class SalidaFormateada02 {
  public static void main(String[] args) {
    System.out.println("  Artículo      Precio/caja    Nº cajas");
    System.out.println("-----------------------------------------");
    System.out.printf("%-10s      %8.2f      %6d\n",  "manzanas", 4.5, 10);
    System.out.printf("%-10s      %8.2f      %6d\n",  "peras", 2.75, 120);
    System.out.printf("%-10s      %8.2f      %6d\n",  "aguacates", 10.0, 6);
  }
}
```

En la casilla `%-10s` se inserta una cadena de caracteres. Esta cadena debe ir entrecomillada. Se reservan 10 caracteres; si la cadena es más pequeña, se rellena con espacios en blanco hasta completar las 10 posiciones. El guión indica que la cadena se debe alinear a la izquierda (`%-10s` alinea a la izquierda y `%10s` alinea a la derecha).

Dentro de la casilla `%8.2f` se inserta un número con decimales. Se reservan 8 caracteres para el número completo, incluyendo la coma decimal y se muestran exactamente 2 decimales.

Por último, la casilla `%6d` muestra por pantalla un número entero colocado en un hueco de 6 posiciones. Si el número ocupa menos, se rellena con espacios en blanco.

El resultado de la ejecución del programa es el siguiente:

```
  Artículo      Precio/caja    Nº cajas
-----------------------------------------
manzanas            4,50          10
peras               2,75         120
aguacates          10,00           6
```

En una cadena formateada se pueden incluir tabuladores con el carácter `\t`, comillas simples con `\'`, comillas dobles con `\"` o icluso la propia barra inclinada con `\\`.

1.6 Ejercicios



Ejercicio 1

Escribe una programa que muestre tu nombre por pantalla.



Ejercicio 2

Modifica el programa anterior para que además se muestre tu dirección y tu número de teléfono. Asegúrate de que los datos se muestran en líneas separadas.



Ejercicio 3

Escribe un programa que muestre por pantalla 10 palabras en inglés junto a su correspondiente traducción al castellano. Las palabras deben estar distribuidas en dos columnas y alineadas a la izquierda.

Ejemplo:

```
computer   ordenador
student    alumno\a
cat        gato
penguin    pingüino
machine    máquina
nature     naturaleza
light      luz
green      verde
book       libro
pyramid    pirámide
```



Ejercicio 4

Escribe un programa que muestre tu horario de clase.



Ejercicio 5

Modifica el programa anterior añadiendo colores. Puedes mostrar cada asignatura de un color diferente.



Ejercicio 6

Escribe un programa que pinte por pantalla una pirámide rellena a base de asteriscos. La base de la pirámide debe estar formada por 9 asteriscos.

Ejemplo:

```
    *
   ***
  *****
 *******
*********
```



Ejercicio 7

Igual que el programa anterior, pero esta vez la pirámide estará hueca (se debe ver únicamente el contorno hecho con asteriscos).

Ejemplo:

```
    *
   * *
  *   *
 *     *
*********
```



Ejercicio 8

Igual que el programa anterior, pero esta vez la pirámide debe aparecer invertida, con el vértice hacia abajo.

Ejemplo:

```
*********
 *     *
  *   *
   * *
    *
```



Ejercicio 9

Escribe un programa que pinte por pantalla alguna escena - el campo, la habitación de una casa, un aula, etc. - o algún objeto animado o inanimado - un coche, un gato, una taza de café, etc. Ten en cuenta que puedes utilizar caracteres como *, +, <, #, @, etc. o incluso caracteres Unicode. ¡Échale imaginación!

Ejemplo:





Ejercicio 10

Mejora el ejercicio anterior añadiéndole colores.

Ejemplo:



2. Variables

2.1 Definición y tipos de variables

Una variable es un contenedor de información. Imagina una variable como una cajita con una etiqueta que indica su nombre, una cajita en la que se puede introducir un valor. Las variables pueden almacenar valores enteros, números decimales, caracteres, cadenas de caracteres (palabras o frases), etc. El contenido de las variables puede cambiar durante la ejecución del programa, de ahí viene el nombre de “variable”.

Java es un lenguaje fuertemente tipado, es decir, es necesario declarar todas las variables que utilizará el programa, indicando siempre el nombre y el tipo de cada una.

El nombre que se le da a una variable es muy importante; intenta usar siempre nombres significativos que, de alguna forma, identifiquen el contenido. Por ejemplo, si usas una variable para almacenar el volumen de una figura, una buena opción sería llamarla `volumen`; si tienes una variable que almacena la edad de una persona, lo mejor es llamarla `edad`, y así sucesivamente.

Un programa se lee y se depura mucho mejor si los nombres de las variables se han elegido de forma inteligente.

¿Podrías averiguar qué hace la siguiente línea de código?

```
x = vIp3 * Weerty - zxc;
```

Ahora observa el mismo código pero con otros nombres de variables.

```
precioTotal = cantidad * precio - descuento;
```

Se entiende mucho mejor ¿verdad? No me cansaré de repetir lo importante que es nombrar correctamente las variables. Piensa que un programador pasa mucho más tiempo leyendo código que programando. Leer código fuente en Java bien escrito debería ser casi tan fácil y natural como leer una novela.

Escribiremos los nombres de variables en formato *lowerCamelCase*. La primera letra se escribe en minúscula y, a continuación, si se utiliza más de una palabra, cada una de ellas empezaría con mayúscula. Por ejemplo, `edadMin` es un buen nombre para una variable que almacena la edad mínima a la que se puede acceder a un sitio web. Observa que hemos usado una mayúscula para diferenciar dos partes (edad y Min). Puede que en algún libro también encuentres nombres de variables con el carácter

de subrayado (_) de tal forma que el nombre de la variable sería `edad_min`, pero como hemos comentado anteriormente, en este libro nos ceñimos al estándar de Google[1], que exige el formato *lowerCamelCase*.

No se permiten símbolos como $, %, @, +, -, etc. Puedes usar números en los nombres de variables pero nunca justo al principio; `5x` no es un nombre válido pero `x5` sí lo es.

No se debe poner una letra mayúscula al comienzo del nombre de una variable para no confundirla con una clase (los nombres de las clases comienzan por mayúscula).



Nombres de variables

Los nombres de las variables deben ser **significativos**, es decir, deben indicar perfectamente qué información contienen.

Enteros (`int` y `long`)

Las variables que van a contener números enteros se declaran con `int`. Veamos un ejemplo.

```java
/**
 * Uso de variables enteras
 *
 * @author Luis J. Sánchez
 */
public class VariablesEnteras {
  public static void main(String[] args) {
    int x; // Declara la variable x como entera

    x = 5; // Asigna el valor 5 a la variable x

        // Muestra el valor de x
    System.out.println("El valor actual de mi variable es " + x);

    x = 7; // Asigna el valor 7 a la variable x

    // Muestra de nuevo el valor de x
    System.out.println("y ahora es " + x);
  }
}
```

Si pretendemos almacenar valores muy grandes en una variable, usaremos el tipo `long` en lugar de `int`.

[1] https://google.github.io/styleguide/javaguide.html

Números decimales (`double` y `float`)

Usamos los tipos `double` o `float` cuando queremos (o esperamos) almacenar números con decimales en las variables. La diferencia está en la precisión, las variables de tipo `double` tienen mayor precisión que las de tipo `float`.

Ciñéndonos al estándar de Google, no daremos por válidas definiciones de variables como la siguiente (aunque funcionaría sin ningún problema).

```
double x, y;
```

Cada variable se debe definir en una línea diferente. Lo siguiente sí sería correcto.

```
double x;
double y;
```

A continuación tienes un ejemplo completo.

```
/**
 * Uso de variables que contienen números decimales
 *
 * @author Luis J. Sánchez
 */
public class VariablesConDecimales {
  public static void main(String[] args) {
    double x; // Se declaran las variables x e y
    double y; // de tal forma que puedan almacenar decimales.

    x = 7;
    y = 25.01;

    System.out.println(" x vale " + x);
    System.out.println(" y vale " + y);
  }
}
```

Como puedes ver, también se pueden almacenar números enteros en variables de tipo `double`.

Cadenas de caracteres (`String`)

Las cadenas de caracteres se utilizan para almacenar palabras y frases. Todas las cadenas de caracteres deben ir entrecomilladas mediante el símbolo de comillas dobles (“).

```java
/**
 * Uso del tipo String
 *
 * @author Luis J. Sánchez
 */
public class UsoDeStrings {
  public static void main(String[] args) {
    String miPalabra = "cerveza";
    String miFrase = "¿dónde está mi cerveza?";

    System.out.println("Una palabra que uso con frecuencia: " + miPalabra);
    System.out.println("Una frase que uso a veces: " + miFrase);
  }
}
```

Un cadena de caracteres puede contener cero (cadena vacía) o más caracteres. Nos podríamos encontrar con un código como el siguiente:

```java
String cadenaInicial = "";
```

En un principio, parece absurdo disponer de una cadena de caracteres que no contiene ningún carácter, pero más adelante verás la utilidad. Por ejemplo, imagina que estás programando el juego del ahorcado y necesitas tener una cadena donde vayas guardando todas las letras que va probando el jugador; en este caso, cuando empieza el juego, no se ha probado ninguna letra, por tanto, esa cadena estará inicialmente vacía y posteriormente se irá rellenando.

Como puedes ver, en una cadena de caracteres se pueden almacenar signos de puntuación, espacios y letras con tildes.

Puede que te estés preguntando ¿por qué los tipos `int`, `long`, `double` y `float` empiezan por minúscula y `String` empieza por mayúscula? La respuesta es que `String` es en realidad una clase, no un tipo primitivo. Veremos las clases en profundidad en el capítulo 9 que trata sobre la programación orientada a objetos.

Caracteres (`char`)

Un carácter suelto como una letra o un signo de puntuación se puede almacenar en una variable de tipo `char`. El carácter debe ir entrecomillado utilizando las comillas simples (‘).

Hay que tener en cuenta que no es lo mismo `"a"` que `'a'`. Aunque el contenido en ambos casos es la letra “a”, lo primero es una cadena de caracteres y lo segundo es un carácter. En algunos lenguajes de programación se pueden usar indistintamente las comillas simples y las dobles pero en Java tienen un significado muy distinto.

```
/**
 * Uso del tipo char
 *
 * @author Luis J. Sánchez
 */
public class UsoDeChar {
  public static void main(String[] args) {
    char letra1 = 'c';
    char letra2 = 'a';
    char letra3 = 's';
    char letra4 = 'a';

    System.out.println("letra1: " + letra1);
    System.out.println("letra3: " + letra3);
    System.out.println("todas las letras juntas: " + letra1 + letra2 + letra3 + letra4);
  }
}
```

Veamos un comportamiento curioso del tipo `char`.

```
/**
 * Correspondencia entre los tipos char e int
 *
 * @author Luis J. Sánchez
 */
public class CharComoInt {
  public static void main(String[] args) {
    char letra1 = 'a';
    char letra2 = 'b';

    System.out.println(letra1);
    System.out.println(letra2);
    System.out.println(letra1 + letra2);
    System.out.println(letra1 + "" + letra2);
  }
}
```

El resultado del programa anterior se muestra a continuación.

```
a
b
195
ab
```

Existe una correspondencia entre los tipos `char` e `int` de tal forma que la suma de dos caracteres se considera la suma de dos enteros. En el programa anterior, la suma

`letra1 + letra2` en realidad es la suma de los códigos ASCII de la letra "a" y de la letra "b" que son el 97 y el 98 respectivamente. En Java se podrían escribir cosas como `'a' + 7` sin que diera ningún error. Fíjate que en el ejemplo, para mostrar el contenido de las variables `letra1` y `letra2` de forma consecutiva se recurre al truco de pintar en medio una cadena de caracteres vacía.

2.2 Resumen de tipos primitivos

Los tipos `int`, `long`, `double` y `float` vistos anteriormente son los llamados "primitivos". Estos tipos forman parte del propio lenguaje y están disponibles desde las primeras versiones. En Java hay 8 tipos primitivos.

TIPO	DESCRIPCIÓN	TAMAÑO	EJEMPLO
boolean	verdadero o falso	1 bit	boolean abierto = true;
byte	número entero	8 bits	byte repeticiones = 22;
char	carácter	16 bits	char letra = 'a';
short	número entero	16 bits	short pantalones = 22;
int	número entero	32 bits	int asistentes = 22;
long	número entero	64 bits	long poblacion = 22L;
float	número con decimales	32 bits	float nota = 9.5f;
double	número con decimales	64 bits	double precio = 22.55d;

2.3 Operadores artiméticos

En Java se puede operar con las variables de una forma muy parecida a como se hace en matemáticas. Los operadores aritméticos de Java son los siguientes:

OPERADOR	NOMBRE	EJEMPLO	DESCRIPCIÓN
+	suma	20 + x	suma dos números
-	resta	a - b	resta dos números
*	multiplicación	10 * 7	multiplica dos números
/	división	altura / 2	divide dos números
%	resto (módulo)	5 % 2	resto de la división entera
++	incremento	a++	incrementa en 1 el valor de la variable
- -	decremento	a- -	decrementa en 1 el valor de la variable

A continuación tienes un programa que ilustra el uso de los operadores aritméticos.

```java
/**
 * Uso de los operadores aritméticos
 *
 * @author Luis J. Sánchez
 */
public class UsoDeOperadoresAritmeticos {
  public static void main(String[] args) {
    int x;
    x = 100;

    System.out.println(x + " " + (x + 5) + " " + (x - 5));
    System.out.println((x * 5) + " " + (x / 5) + " " + (x % 5));
  }
}
```

2.4 Asignación de valores a variables

La sentencia de asignación se utiliza para dar un valor a una variable. En Java (y en la mayoría de lenguajes de programación) se utiliza el símbolo igual (=) para este cometido. Es importante recalcar que una asignación no es una ecuación. Por ejemplo `x = 7 + 1` es una asignación en la cual se evalúa la parte derecha `7 + 1`, y el resultado de esa evaluación se almacena en la variable que se coloque a la izquierda del igual, es decir, en la `x`, o lo que es lo mismo, el número 8 se almacena en `x`. La sentencia `x + 1 = 23 * 2` no es una asignación válida ya que en el lado izquierdo debemos tener únicamente un nombre de variable.

Veamos un ejemplo con algunas operaciones y asignaciones.

```java
/**
 * Operaciones y asignaciones
 *
 * @author Luis J. Sánchez
 */
public class Asignaciones {
  public static void main(String[] args) {
    int x = 2;
    int y = 9;

    int sum = x + y;
    System.out.println("La suma de mis variables es " + sum);

    int mul = x * y;
    System.out.println("La multiplicación de mis variables es " + mul);
```

```
  }
}
```

2.5 Conversión de tipos (*casting*)

En ocasiones es necesario convertir una variable (o una expresión en general) de un tipo a otro. Simplemente hay que escribir entre paréntesis el tipo que se quiere obtener. Experimenta con el siguiente programa y observa los diferentes resultados que se obtienen.

```
/**
 * Conversión de tipos
 *
 * @author Luis J. Sánchez
 */
public class ConversionDeTipos {
  public static void main(String[] args) {
    int x = 2;
    int y = 9;
    double division;

    division = (double) y / (double) x;

    // Descomenta la siguiente línea y observa cómo cambia el resultado.
    // division = y / x;

    System.out.println("El resultado de la división es " + division);
  }
}
```

Primero tenemos

```
division = (double) y / (double) x
```

`(double) y` convierte el valor de `y` en un `double`, es decir, en un número con decimales y `(double) x` hace lo propio con `x`, por tanto, la operación de división quedaría como `9.0 / 2.0`. Al tratarse de una división entre números decimales, Java entiende que el resultado debe tener también decimales y usa toda la precisión disponible en el tipo. El resultado que se guarda en la variable `division` es `4.5` que es el valor correcto.

Descomentando la línea 15, tendríamos

```
division = y / x
```

Ahora la división es `9 / 2`. Java ve una división entre dos números enteros, así que el resultado que ofrece es otro número entero después de despreciar los decimales si los hubiera. El resultado que da Java es la división entera, o sea, `4`. El siguiente paso es una asignación, tendríamos `division = 4`. Como la variable `division` es de tipo `double`, el valor `4` se guarda en realidad como `4.0`; es curioso ¿verdad?



Usa el *casting* para conservar los decimales en las divisiones con enteros

La división entre dos números enteros es otro número entero. Para conservar los decimales de la división hay que hacer un *casting* a `float` o `double`.

2.6 Ejercicios



Ejercicio 1

Escribe un programa en el que se declaren las variables enteras `x` e `y`. Asígnales los valores 144 y 999 respectivamente. A continuación, muestra por pantalla el valor de cada variable, la suma, la resta, la división y la multiplicación.



Ejercicio 2

Crea la variable `nombre` y asígnale tu nombre completo. Muestra su valor por pantalla de tal forma que el resultado del programa sea el mismo que en el ejercicio 1 del capítulo 1.



Ejercicio 3

Crea las variables `nombre`, `direccion` y `telefono` y asígnales los valores correspondientes. Muestra los valores de esas variables por pantalla de tal forma que el resultado del programa sea el mismo que en el ejercicio 2.



Ejercicio 4

Realiza un conversor de euros a pesetas. La cantidad en euros que se quiere convertir deberá estar almacenada en una variable.



Ejercicio 5

Realiza un conversor de pesetas a euros. La cantidad en pesetas que se quiere convertir deberá estar almacenada en una variable.



Ejercicio 6

Escribe un programa que calcule el total de una factura a partir de la base imponible (precio sin IVA). La base imponible estará almacenada en una variable.



Ejercicio 7

Escribe un programa que declare variables de tipo `char` y de tipo `String`. Intenta mostrarlas por pantalla todas juntas en la misma línea y con una sola sentencia de Java (con un solo `println`) ¿es posible?



Ejercicio 8

Escribe un programa que declare 5 variables de tipo `char`. A continuación, crea otra variable como cadena de caracteres y asígnale como valor la concatenación de las anteriores 5 variables. Por último, muestra la cadena de caracteres por pantalla ¿Qué problemas te encuentras? ¿cómo lo has solucionado?

3. Lectura de datos desde teclado

Hemos visto en los capítulos anteriores cómo mostrar información por pantalla y cómo usar variables. Veremos ahora algo muy importante que te permitirá realizar programas algo más funcionales, que tengan una utilidad real; aprenderás a leer la información que introduce un usuario mediante el teclado.

El funcionamiento de casi todos los programas se podría resumir en los siguientes puntos:

1. Entrada de datos desde teclado (o desde cualquier otro dispositivo de entrada)
2. Procesamiento de los datos de entrada para producir un resultado
3. Visualización de los resultados por pantalla

De momento, hemos visto cómo realizar los puntos 2 y 3. Vamos a ver a continuación cómo se recoge la información desde el teclado.

3.1 Lectura de texto

Para recoger datos por teclado usamos `System.console().readLine()`. Cuando llegamos a esta sentencia, el programa se detiene y espera que el usuario introduzca información mediante el teclado. La introducción de datos termina con la pulsación de la tecla INTRO. Una vez que el usuario presiona INTRO, todo lo que se ha tecleado se almacena en una variable, en el siguiente ejemplo esa variable es `nombre`.

```
/**
 * Lectura de datos desde teclado
 *
 * @author Luis J. Sanchez
 */
public class DimeTuNombre {
  public static void main(String[] args) {
    String nombre;
    System.out.print("Por favor, dime cómo te llamas: ");
    nombre = System.console().readLine();
    System.out.println("Hola " + nombre + ", ¡encantado de conocerte!");
  }
}
```



Mediante `System.console().readLine()` se recoge una línea de texto introducida por teclado.

Cuando se piden datos por teclado es importante que el programa especifique claramente cuál es exactamente la información que requiere por parte del usuario. Date cuenta que este programa muestra el mensaje “`Por favor, dime cómo te llamas:`”. Imagina que se omite este mensaje, en la pantalla aparecería únicamente un cursor parpadeando y el usuario no sabría qué tiene que hacer o qué dato introducir.

También es importante reseñar que los datos introducidos por teclado se recogen como una cadena de caracteres (un `String`).

3.2 Lectura de números

Si en lugar de texto necesitamos datos numéricos, deberemos convertir la cadena introducida en un número con el método adecuado. Como se muestra en el ejemplo, `Integer.parseInt()` convierte el texto introducido por teclado en un dato numérico, concretamente en un número entero.

```java
/**
 * Lectura de datos desde teclado
 *
 * @author Luis J. Sánchez
 */
public class LeeNumeros {
  public static void main(String[] args) {

    String linea;

    System.out.print("Por favor, introduce un número: ");
    linea = System.console().readLine();
    int primerNumero;
    primerNumero = Integer.parseInt( linea );

    System.out.print("introduce otro, por favor: ");
    linea = System.console().readLine();
    int segundoNumero;
    segundoNumero = Integer.parseInt( linea );

    int total;
    total = (2 * primerNumero) + segundoNumero;

    System.out.print("El primer número introducido es " + primerNumero);
    System.out.println(" y el segundo es " + segundoNumero);
```

```
    System.out.print("El doble del primer número más el segundo es ");
    System.out.print(total);
  }
}
```

Este último programa se podría acortar un poco. Por ejemplo, estas dos líneas

```
int total;
total = (2 * primerNumero) + segundoNumero;
```

se podrían quedar en una sola línea

```
int total = (2 * primerNumero) + segundoNumero;
```

De igual modo, estas tres líneas

```
linea = System.console().readLine();
int primerNumero;
primerNumero = Integer.parseInt( linea );
```

también se podrían reducir a una sola tal que así

```
int primerNumero = Integer.parseInt( System.console().readLine() );
```

Es muy importante que el código de nuestros programas sea limpio y legible. A veces, abreviando demasiado el código se hace más difícil de leer; es preferible tener unas líneas de más y que el código se entienda bien a tener un código muy compacto pero menos legible.

3.3 La clase `Scanner`

El método `System.console().readLine()` funciona bien en modo consola (en una ventana de terminal) pero puede provocar problemas cuando se trabaja con IDEs como **Eclipse**, **Netbeans**, **JavaEdit**, etc. Para evitar estos problemas puedes usar la clase `Scanner` cuando necesites recoger datos desde teclado. La clase `Scanner` funciona tanto en entornos integrados como en una ventana de terminal.

```java
import java.util.Scanner;

/**
 * Lectura de datos desde teclado usando la clase Scanner
 *
 * @author Luis J. Sánchez
 */
public class LeeDatosScanner01 {
  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);

    System.out.print("Introduce tu nombre: ");
    String nombre = s.nextLine();

    System.out.print("Introduce tu edad: ");
    int edad = Integer.parseInt(s.nextLine());

    System.out.println("Tu nombre es " + nombre + " y tu edad es " + edad);
  }
}
```

Fíjate que en el programa anterior la sentencia

```java
s.nextLine()
```

sería el equivalente a

```java
System.console().readLine()
```

Mediante el uso de la clase `Scanner` es posible leer varios datos en una misma línea. En el programa anterior se pedía un nombre y una edad, en total dos datos que había que introducir en líneas separadas. Observa cómo en el siguiente ejemplo se piden esos dos datos en una sola línea y separados por un espacio.

```java
import java.util.Scanner;

/**
 * Lectura de datos desde teclado usando la clase Scanner
 *
 * @author Luis J. Sánchez
 */
public class LeeDatosScanner02 {
  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
```

```
    System.out.print("Introduce tu nombre y tu edad separados por un espacio: ");
    String nombre = s.next();
    int edad = s.nextInt();

    System.out.println("Tu nombre es " + nombre + " y tu edad es " + edad);
  }
}
```

Fíjate cómo se ha utilizado `s.next()` para leer una cadena de caracteres y `s.nextInt()` para leer un número entero, todo ello en la misma línea.

El siguiente programa de ejemplo calcula la media de tres números decimales. Para leer cada uno de los números en la misma línea se utiliza `s.nextDouble`.

```
import java.util.Scanner;

/**
 * Lectura de datos desde teclado usando la clase Scanner
 *
 * @author Luis J. Sánchez
 */
public class LeeDatosScannerMedia {
  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);

    System.out.print("Introduce tres números (pueden contener decimales) separados por espacio\
s: ");
    double x1 = s.nextDouble();
    double x2 = s.nextDouble();
    double x3 = s.nextDouble();

    double media = (x1 + x2 + x3) / 3;

    System.out.println("La media de esos tres números es " + media);
  }
}
```



Recuerda que **¡a programar se aprende programando!** Es muy importante que pruebes los ejemplos - si los modificas y experimentas, todavía mejor - y que realices los ejercicios. Para aprender a programar no basta con leer el libro y entenderlo, es una condición necesaria pero no suficiente. Igual que para ser piloto hacen falta muchas horas de vuelo, para ser programador hacen falta muchas horas picando código, probando, corrigiendo errores… ¡Ánimo! ¡El esfuerzo merece la pena!

3.4 Ejercicios



Ejercicio 1

Realiza un programa que pida dos números y que luego muestre el resultado de su multiplicación.



Ejercicio 2

Realiza un conversor de euros a pesetas. La cantidad de euros que se quiere convertir debe ser introducida por teclado.



Ejercicio 3

Realiza un conversor de pesetas a euros. La cantidad de pesetas que se quiere convertir debe ser introducida por teclado.



Ejercicio 4

Escribe un programa que sume, reste, multiplique y divida dos números introducidos por teclado.



Ejercicio 5

Escribe un programa que calcule el área de un rectángulo.



Ejercicio 6

Escribe un programa que calcule el área de un triángulo.



Ejercicio 7

Escribe un programa que calcule el total de una factura a partir de la base imponible.



Ejercicio 8

Escribe un programa que calcule el salario semanal de un empleado en base a las horas trabajadas, a razón de 12 euros la hora.



Ejercicio 9

Escribe un programa que calcule el volumen de un cono según la fórmula $V = \frac{1}{3}\pi r^2 h$



Ejercicio 10

Realiza un conversor de Mb a Kb.



Ejercicio 11

Realiza un conversor de Kb a Mb.



Ejercicio 12

Realiza un programa que calcule la nota que hace falta sacar en el segundo examen de la asignatura **Programación** para obtener la media deseada. Hay que tener en cuenta que la nota del primer examen cuenta el 40% y la del segundo examen un 60%.

Ejemplo 1:

```
Introduce la nota del primer examen: 7
¿Qué nota quieres sacar en el trimestre? 8.5
Para tener un 8.5 en el trimestre necesitas sacar un 9.5 en el segundo examen.
```

Ejemplo 2:

```
Introduce la nota del primer examen: 8
¿Qué nota quieres sacar en el trimestre? 7
Para tener un 7 en el trimestre necesitas sacar un 6.33 en el segundo examen.
```

4. Sentencia condicional (`if` y `switch`)

Una sentencia condicional permite al programa bifurcar el flujo de ejecución de instrucciones dependiendo del valor de una expresión.

4.1 Sentencia `if`

La sentencia `if` permite la ejecución de una serie de instrucciones en función del resultado de una expresión lógica. El resultado de evaluar una expresión lógica es siempre verdadero (`true`) o falso (`false`). Es muy simple, en lenguaje natural sería algo como ”si esta condición es verdadera entonces haz esto, sino haz esto otro”.

El formato de la sentencia `if` es el siguiente:

```
if (condición) {

    instrucciones a ejecutar si la condición es verdadera

} else {

    instrucciones a ejecutar si la condición es falsa

}
```

A continuación se muestra un ejemplo del uso de la sentencia `if`.

```java
/**
 * Sentencia if
 *
 * @author Luis J. Sánchez
 */
public class ComparacionCadena {
  public static void main(String[] args) {
    String miFruta = "naranja";

    if ("naranja".equals(miFruta)) {
      System.out.println("iguales");
    } else {
      System.out.println("distintas");
    }
```

```
  }
}
```

¿Podrías adivinar cuál es el resultado del programa? Efectivamente, se muestra la palabra "iguales" ya que la comparación `"naranja".equals(miFruta)` devuelve `true` y, por tanto, se ejecuta el primer bloque te código, o sea, `System.out.println("iguales");`.

Veamos ahora un ejemplo algo más interesante. Se trata de una especie de juego al estilo de "La ruleta de la fortuna" en el que el usuario debe responder a la pregunta que le hace el ordenador.

```
/**
 * Sentencia if
 *
 * @author Luis J. Sánchez
 */
public class SentenciaIf01 {
  public static void main(String[] args) {
    System.out.print("¿Cuál es la capital de Kiribati? ");
    String respuesta = System.console().readLine();

    if (respuesta.equals("Tarawa")) {
      System.out.println("¡La respuesta es correcta!");
    } else {
      System.out.println("Lo siento, la respuesta es incorrecta.");
    }
  }
}
```

En el programa se le pregunta al usuario cuál es la capital de Kiriwati. La respuesta introducida por el usuario se almacena en la variable `respuesta`. A continuación viene la sentencia condicional `if`.

```
if (respuesta.equals("Tarawa"))
```

Llegado a este punto, el programa evalúa la expresión `respuesta.equals("Tarawa")`. Observa que para comparar dos cadenas de caracteres se utiliza `equals()`. Imaginemos que el usuario ha introducido por teclado `Madrid`; entonces la expresión `"Madrid".equals("Tarawa")` daría como resultado `false` (falso).

Si la expresión hubiera dado como resultado `true` (verdadero), se ejecutaria la línea

```
System.out.println("¡La respuesta es correcta!");
```

pero no es el caso, el resultado de la expresión ha sido `false` (falso), todo el mundo sabe que la capital de Kiriwati no es Madrid, por tanto se ejecutaría la línea

```
System.out.println("Lo siento, la respuesta es incorrecta.");
```

Vamos a ver otro ejemplo, esta vez con números. El usuario introducirá un número por teclado y el programa dirá si se trata de un número positivo o negativo.

```
/**
 * Sentencia if
 *
 * @author Luis J. Sánchez
 */
public class SentenciaIf02 {
  public static void main(String[] args) {
    System.out.print("Por favor, introduce un número entero: ");
    String linea = System.console().readLine();
    int x = Integer.parseInt( linea );

    if (x < 0) {
      System.out.println("El número introducido es negativo.");
    } else {
      System.out.println("El número introducido es positivo.");
    }
  }
}
```

El siguiente bloque de código

```
if (x < 0)
  System.out.println("El número introducido es negativo.");
else
  System.out.println("El número introducido es positivo.");
```

compilaría y funcionaría sin problemas - fíjate que hemos quitado las llaves - ya que antes y después del `else` hay una sola sentencia y en estos casos no es obligatorio poner llaves. Sin embargo, nosotros siempre usaremos llaves, es una exigencia del estándar de Google al que nos ceñimos en este manual.



Llaves egipcias (egyptian brackets)

Fíjate en la manera de colocar las llaves dentro del código de un programa en Java. La llave de apertura de bloque se coloca justo al final de la línea y la llave de cierre va justo al principio de la línea. Se llaman llaves egipcias[1] por la similitud entre las llaves y las manos de los egipcios que aparecen en los papiros.

[1] http://blog.codinghorror.com/new-programming-jargon/

4.2 Operadores de comparación

En el ejemplo anterior, usamos el operador `<` en la comparación `if (x < 0)` para saber si la variable `x` es menor que cero. Hay más operadores de comparación, en la tabla 4.2.1 se muestran todos.

Tabla 4.2.1. Operadores de comparación.

OPERADOR	NOMBRE	EJEMPLO	DESCRIPCIÓN
==	igual	a == b	a es igual a b
!=	distinto	a != b	a es distinto de b
<	menor que	a < b	a es menor que b
>	mayor que	a > b	a es mayor que b
<=	menor o igual que	a < = b	a es menor o igual que b
>=	mayor o igual que	a > = b	a es mayor o igual que b

El siguiente ejemplo muestra el uso de uno de estos operadores, concretamente de `>=` (mayor o igual). El usuario introduce una nota; si esta nota es **mayor o igual a 5** se le mostrará un mensaje diciendo que ha aprobado y en caso de que no se cumpla la condición se mostrará un mensaje diciendo que está suspenso.

```
/**
 * Sentencia if
 *
 * @author Luis J. Sánchez
 */
public class SentenciaIf03 {
  public static void main(String[] args) {
    System.out.print("¿Qué nota has sacado en el último examen? ");
    String line = System.console().readLine();
    double nota = Double.parseDouble( line );

    if (nota >= 5) {
      System.out.println("¡Enhorabuena!, ¡has aprobado!");
    } else {
      System.out.println("Lo siento, has suspendido.");
    }
  }
}
```

4.3 Operadores lógicos

Los operadores de comparación se pueden combinar con los operadores lógicos. Por ejemplo, si queremos saber si la variable `a` es mayor que `b` y además es menor que `c`, escribiríamos `if ((a > b) && (a < c))`. En la siguiente tabla se muestran los operadores lógicos de Java:

Tabla 4.3.1. Operadores lógicos.

OPERADOR	NOMBRE	EJEMPLO	DEVUELVE VERDADERO CUANDO…
&&	y	(7 > 2) && (2 < 4)	las dos condiciones son verdaderas
||	o	(7 > 2) || (2 < 4)	al menos una de las condiciones es verdadera
!	no	!(7 > 2)	la condición es falsa

Vamos a ver cómo funcionan los operadores lógicos con un ejemplo. Mediante `if ((n < 1) || (n > 100))` se pueden detectar los números que no están en el rango de 1 a 100; literalmente sería “si n es menor que 1 o n es mayor que 100”.

```java
/**
 * Operadores lógicos
 *
 * @author Luis J. Sánchez
 */
public class OperadoresLogicos01 {
  public static void main(String[] args) {
    System.out.println("Adivina el número que estoy pensando.");
    System.out.print("Introduce un número entre el 1 y el 100: ");
    String linea = System.console().readLine();
    int n = Integer.parseInt( linea );

    if ((n < 1) || (n > 100)) {
      System.out.println("El número introducido debe estar en el intervalo 1 - 100.");
      System.out.print("Tienes otra oportunidad, introduce un número: ");
      linea = System.console().readLine();
      n = Integer.parseInt( linea );
    }

    if (n == 24) {
      System.out.println("¡Enhorabuena!, ¡has acertado!");
    } else {
      System.out.println("Lo siento, ese no es el número que estoy pensando.");
```

```
    }
  }
}
```

El comportamiento de los operadores lógicos se muestra en la siguiente tabla de verdad (tabla 4.3.2) donde “V” significa “verdadero” y “F” significa “falso”.

Tabla 4.3.2. Tabla de verdad de los peradores lógicos.

A	B	A && B	A || B	!A	!B
V	V	V	V	F	F
V	F	F	V	F	V
F	V	F	V	V	F
F	F	F	F	V	V

En el siguiente programa puedes ver el uso de operadores lógicos combinado con operadores relacionales (operadores de comparación). Intenta adivinar cuál será el resultado mirando el código.

```
/**
 * Operadores lógicos y relacionales
 *
 * @author Luis J. Sánchez
 */
public class OperadoresLogicos02 {
  public static void main(String[] args) {
    boolean a = true;
    boolean b = false;
    System.out.println("a = " + a);
    System.out.println("b = " + b);
    System.out.println("a && b = " + (a && b));
    System.out.println("a || b = " + (a || b));
    System.out.println("!a = " + !a);
    System.out.println("a || (6 > 10) = " + (a || (6 > 10)));
    System.out.println("((4 <= 4) || false) && (!a) = " + (((4 <= 4) || false) && (!a)));
  }
}
```

4.4 Sentencia switch (selección múltiple)

A veces es necesario comparar el valor de una variable con una serie de valores concretos. La selección múltiple es muy parecida (aunque no es exactamente igual) a una secuencia de varias sentencias if.

El formato de switch es el que se muestra a continuación. En lenguaje natural sería algo así como "Si variable vale valor1 entonces entra por case valor1:, si variable vale valor2 entonces entra por case valor2:,… si variable no vale ninguno de los valores que hay en los distintos case entonces entra por default:.

```
switch(variable) {
  case valor1:
    sentencias
    break;

  case valor2:
    sentencias
    break;
    .
    .
    .

  default:
    sentencias
    }
```

A continuación tienes un ejemplo completo en Java. Se pide al usuario un número de mes y el programa da el nombre del mes que corresponde a ese número.

```
/**
 * Sentencia múltiple (switch)
 *
 * @author Luis José Sánchez
 */
public class SentenciaSwitch {
  public static void main(String[] args) {

    System.out.print("Por favor, introduzca un numero de mes: ");
    int mes = Integer.parseInt(System.console().readLine());

    String nombreDelMes;

    switch (mes) {
      case 1:
```

```
      nombreDelMes = "enero";
      break;
    case 2:
      nombreDelMes = "febrero";
      break;
    case 3:
      nombreDelMes = "marzo";
      break;
    case 4:
      nombreDelMes = "abril";
      break;
    case 5:
      nombreDelMes = "mayo";
      break;
    case 6:
      nombreDelMes = "junio";
      break;
    case 7:
      nombreDelMes = "julio";
      break;
    case 8:
      nombreDelMes = "agosto";
      break;
    case 9:
      nombreDelMes = "septiembre";
      break;
    case 10:
      nombreDelMes = "octubre";
      break;
    case 11:
      nombreDelMes = "noviembre";
      break;
    case 12:
      nombreDelMes = "diciembre";
      break;
    default:
      nombreDelMes = "no existe ese mes";
  }

  System.out.println("Mes " + mes + ": " + nombreDelMes);
 }
}
```

Observa que es necesario introducir un `break` después de la asignación de la variable `nombreDelMes`. En caso de no encontrarse el `break`, el programa continúa la ejecución en la línea siguiente.

El bloque que corresponde al `default` se ejecuta cuando la variable no coincide con ninguno de los valores de los `case`. Escribiremos siempre el `default` al final de la sentencia `switch` aunque no sea necesario.

La sentencia `switch` se utiliza con frecuencia para crear menús.

```
/**
 * Ejemplo de un menú con switch
 *
 * @author Luis José Sánchez
 */
public class MenuConSwitch {
  public static void main(String[] args) {

    System.out.println(" CÁLCULO DE ÁREAS");
    System.out.println(" ----------------");
    System.out.println(" 1. Cuadrado");
    System.out.println(" 2. Rectángulo");
    System.out.println(" 3. Triángulo");
    System.out.print("\n Elija una opción (1-3): ");

    int opcion = Integer.parseInt(System.console().readLine());

    double lado;
    double base;
    double altura;

    switch (opcion) {
      case 1:
        System.out.print("\nIntroduzca el lado del cuadrado en cm: ");
        lado = Double.parseDouble(System.console().readLine());
        System.out.println("\nEl área del cuadrado es " + (lado * lado) + " cm2");
        break;

      case 2:
        System.out.print("\nIntroduzca la base del rectángulo en cm: ");
        base = Double.parseDouble(System.console().readLine());
        System.out.print("Introduzca la altura del rectángulo en cm: ");
        altura = Double.parseDouble(System.console().readLine());
        System.out.println("El área del rectángulo es " + (base * altura) + " cm2");
        break;

      case 3:
        System.out.print("\nIntroduzca la base del triángulo en cm: ");
        base = Double.parseDouble(System.console().readLine());
        System.out.print("Introduzca la altura del triángulo en cm: ");
        altura = Double.parseDouble(System.console().readLine());
```

```
        System.out.println("El área del triángulo es " + ((base * altura) / 2) + " cm2");
        break;

      default:
        System.out.print("\nLo siento, la opción elegida no es correcta.");
    }
  }
}
```

4.5 Ejercicios



Ejercicio 1

Escribe un programa que pida por teclado un día de la semana y que diga qué asignatura toca a primera hora ese día.



Ejercicio 2

Realiza un programa que pida una hora por teclado y que muestre luego buenos días, buenas tardes o buenas noches según la hora. Se utilizarán los tramos de 6 a 12, de 13 a 20 y de 21 a 5. respectivamente. Sólo se tienen en cuenta las horas, los minutos no se deben introducir por teclado.



Ejercicio 3

Escribe un programa en que dado un número del 1 a 7 escriba el correspondiente nombre del día de la semana.



Ejercicio 4

Vamos a ampliar uno de los ejercicios de la relación anterior para considerar las horas extras. Escribe un programa que calcule el salario semanal de un trabajador teniendo en cuenta que las horas ordinarias (40 primeras horas de trabajo) se pagan a 12 euros la hora. A partir de la hora 41, se pagan a 16 euros la hora.

Ejemplo 1:

```
Por favor, introduzca el número de horas trabajadas durante la semana: 36
El sueldo semanal que le corresponde es de 432 euros
```

Ejemplo 2:

```
Por favor, introduzca el número de horas trabajadas durante la semana: 40
El sueldo semanal que le corresponde es de 480 euros
```

Ejemplo 3:

```
Por favor, introduzca el número de horas trabajadas durante la semana: 55
El sueldo semanal que le corresponde es de 720 euros
```



Ejercicio 5

Realiza un programa que resuelva una ecuación de primer grado (del tipo $ax+b = 0$).

Ejemplo 1:

```
Este programa resuelve ecuaciones de primer grado del tipo ax + b = 0
Por favor, introduzca el valor de a: 2
Ahora introduzca el valor de b: 1
x = -0.5
```

Ejemplo 2:

```
Este programa resuelve ecuaciones de primer grado del tipo ax + b = 0
Por favor, introduzca el valor de a: 0
Ahora introduzca el valor de b: 7
Esa ecuación no tiene solución real.
```



Ejercicio 6

Realiza un programa que calcule el tiempo que tardará en caer un objeto desde una altura h. Aplica la fórmula $t = \sqrt{\frac{2h}{g}}$ siendo $g = 9.81 m/s^2$



Ejercicio 7

Realiza un programa que calcule la media de tres notas.



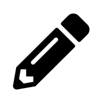
Ejercicio 8

Amplía el programa anterior para que diga la nota del boletín (insuficiente, suficiente, bien, notable o sobresaliente).



Ejercicio 9

Realiza un programa que resuelva una ecuación de segundo grado (del tipo $ax^2 + bx + c = 0$).



Ejercicio 10

Escribe un programa que nos diga el horóscopo a partir del día y el mes de nacimiento.



Ejercicio 11

Escribe un programa que dada una hora determinada (horas y minutos), calcule los segundos que faltan para llegar a la medianoche.



Ejercicio 12

Realiza un minicuestionario con 10 preguntas tipo test sobre las asignaturas que se imparten en el curso. Cada pregunta acertada sumará un punto. El programa mostrará al final la calificación obtenida. Pásale el minicuestionario a tus compañeros y pídeles que lo hagan para ver qué tal andan de conocimientos en las diferentes asignaturas del curso.



Ejercicio 13

Escribe un programa que ordene tres números enteros introducidos por teclado.



Ejercicio 14

Realiza un programa que diga si un número introducido por teclado es par y/o divisible entre 5.



Ejercicio 15

Escribe un programa que pinte una pirámide rellena con un carácter introducido por teclado que podrá ser una letra, un número o un símbolo como *, +, -, $, &, etc. El programa debe permitir al usuario mediante un menú elegir si el vértice de la pirámide está apuntando hacia arriba, hacia abajo, hacia la izquierda o hacia la derecha.



Ejercicio 16

Realiza un programa que nos diga si hay probabilidad de que nuestra pareja nos está siendo infiel. El programa irá haciendo preguntas que el usuario contestará con verdadero o falso. Cada pregunta contestada como verdadero sumará 3 puntos. Las preguntas contestadas con falso no suman puntos. A continuación se listan las preguntas del test.

1. Tu pareja parece estar más inquieta de lo normal sin ningún motivo aparente.
2. Ha aumentado sus gastos de vestuario
3. Ha perdido el interés que mostraba anteriormente por ti
4. Ahora se afeita y se asea con más frecuencia (si es hombre) o ahora se arregla el pelo y se asea con más frecuencia (si es mujer)
5. No te deja que mires la agenda de su teléfono móvil
6. A veces tiene llamadas que dice no querer contestar cuando estás tú delante
7. Últimamente se preocupa más en cuidar la línea y/o estar bronceado/a
8. Muchos días viene tarde después de trabajar porque dice tener mucho más trabajo
9. Has notado que últimamente se perfuma más
10. Se confunde y te dice que ha estado en sitios donde no ha ido contigo

A continuación se muestran los mensajes que deberá dar el programa según la puntuación obtenida.

- Puntuación entre 0 y 10: ¡Enhorabuena! tu pareja parece ser totalmente fiel.
- Puntuación entre 11 y 22: Quizás exista el peligro de otra persona en su vida o en su mente, aunque seguramente será algo sin importancia. No bajes la guardia.
- Puntuación entre 22 y 30: Tu pareja tiene todos los ingredientes para estar viviendo un romance con otra persona. Te aconsejamos que indagues un poco más y averigües que es lo que está pasando por su cabeza.



Ejercicio 17

Escribe un programa que diga cuál es la última cifra de un número entero introducido por teclado.



Ejercicio 18

Escribe un programa que diga cuál es la primera cifra de un número entero introducido por teclado. Se permiten números de hasta 5 cifras.



Ejercicio 19

Realiza un programa que nos diga cuántos dígitos tiene un número entero que puede ser positivo o negativo. Se permiten números de hasta 5 dígitos.



Ejercicio 20

Realiza un programa que diga si un número entero positivo introducido por teclado es capicúa. Se permiten números de hasta 5 cifras.



Ejercicio 21

Calcula la nota de un trimestre de la asignatura **Programación**. El programa pedirá las dos notas que ha sacado el alumno en los dos primeros controles. Si la media de los dos controles da un número mayor o igual a 5, el alumno está aprobado y se mostrará la media. En caso de que la media sea un número menor que 5, el alumno habrá tenido que hacer el examen de recuperación que se califica como **apto** o **no apto**, por tanto se debe preguntar al usuario **¿Cuál ha sido el resultado de la recuperación? (apto/no apto)**. Si el resultado de la recuperación es **apto**, la nota será un 5; en caso contrario, se mantiene la nota media anterior.

Ejemplo 1:

```
Nota del primer control: 7
Nota del segundo control: 10
Tu nota de Programación es 8.5
```

Ejemplo 2:

```
Nota del primer control: 6
Nota del segundo control: 3
¿Cuál ha sido el resultado de la recuperación? (apto/no apto): apto
Tu nota de Programación es 5
```

Ejemplo 3:

```
Nota del primer control: 6
Nota del segundo control: 3
¿Cuál ha sido el resultado de la recuperación? (apto/no apto): no apto
Tu nota de Programación es 4.5
```



Ejercicio 22

Realiza un programa que, dado un día de la semana (de lunes a viernes) y una hora (horas y minutos), calcule cuántos minutos faltan para el fin de semana. Se considerará que el fin de semana comienza el viernes a las 15:00h. Se da por hecho que el usuario introducirá un día y hora correctos, anterior al viernes a las 15:00h.



Ejercicio 23

Escribe un programa que calcule el precio final de un producto según su base imponible (precio antes de impuestos), el tipo de IVA aplicado (general, reducido o superreducido) y el código promocional. Los tipos de IVA general, reducido y superreducido son del 21%, 10% y 4% respectivamente. Los códigos promocionales pueden ser **nopro**, **mitad**, **meno5** o **5porc** que significan respectivamente que no se aplica promoción, el precio se reduce a la mitad, se descuentan 5 euros o se descuenta el 5%. El ejercicio se da por bueno si se muestran los valores correctos, aunque los números no estén tabulados.

Ejemplo:

```
Introduzca la base imponible: 25
Introduzca el tipo de IVA (general, reducido o superreducido): reducido
Introduzca el código promocional (nopro, mitad, meno5 o 5porc): mitad
Base imponible       25.00
IVA (10%)             2.50
Precio con IVA       27.50
Cód. promo. (mitad): -13.75
TOTAL                13.75
```



Ejercicio 24

Escribe un programa que genere la nómina (bien desglosada) de un empleado según las siguientes condiciones:

- Se pregunta el cargo del empleado (1 - Prog. junior, 2 - Prog. senior, 3 - Jefe de proyecto), los días que ha estado de viaje visitando clientes durante el mes y su estado civil (1 - Soltero, 2 - Casado).
- El sueldo base según el cargo es de 950, 1200 y 1600 euros según si se trata de un prog. junior, un prog. senior o un jefe de proyecto respectivamente.
- Por cada día de viaje visitando clientes se pagan 30 euros extra en concepto de dietas. Al sueldo neto hay que restarle el IRPF, que será de un 25% en caso de estar soltero y un 20% en caso de estar casado.

Ejemplo:

```
1 - Programador junior
2 - Prog. senior
3 - Jefe de proyecto
Introduzca el cargo del empleado (1 - 3): 2
¿Cuántos días ha estado de viaje visitando clientes? 5
Introduzca su estado civil (1 - Soltero, 2 - Casado): 2

---------------------------------
| Sueldo base           1200,00 |
| Dietas ( 5 viajes)     150,00 |
|-------------------------------|
| Sueldo bruto          1350,00 |
| Retención IRPF (20%)   270,00 |
|-------------------------------|
| Sueldo neto           1080,00 |
---------------------------------
```



Ejercicio 25

La tienda online **BanderaDeEspaña.es** vende banderas personalizadas de la máxima calidad y nos ha pedido hacer un configurador que calcule el precio según el alto y el ancho. El precio base de una bandera es de un céntimo de euro el centímetro cuadrado. Si la queremos con un escudo bordado, el precio se incrementa en 2.50 € independientemente del tamaño. Los gastos de envío son 3.25 €. El IVA ya está incluido en todas las tarifas.

Ejemplo 1:

```
Introduzca la altura de la bandera en cm: 20
Ahora introduzca la anchura: 35
¿Quiere escudo bordado? (s/n): n
Gracias. Aquí tiene el desglose de su compra.
Bandera de   700 cm2:  7,00 €
Sin escudo:            0,00 €
Gastos de envío:       3,25 €
Total:                10,25 €
```

Ejemplo 2:

```
Introduzca la altura de la bandera en cm: 10
Ahora introduzca la anchura: 15
¿Quiere escudo bordado? (s/n): s
Gracias. Aquí tiene el desglose de su compra.
Bandera de   150 cm2:  1,50 €
Con escudo:            2,50 €
Gastos de envío:       3,25 €
Total:                 7,25 €
```



Ejercicio 26

Realiza un programa que calcule el precio de unas entradas de cine en función del número de personas y del día de la semana. El precio base de una entrada son 8 euros. El miércoles (día del espectador), el precio base es de 5 euros. Los jueves son el día de la pareja, por lo que la entrada para dos cuesta 11 euros. Con la tarjeta CineCampa se obtiene un 10% de descuento. Si un jueves, un grupo de 6 personas compran entradas, el precio total sería de 33 euros ya que son 3 parejas; pero si es un grupo de 7, pagarán 3 entradas de pareja más 1 individual que son 41 euros (33 + 8).

Ejemplo 1:

```
Venta de entradas CineCampa
Número de entradas: 4
```

```
Día de la semana: martes
¿Tiene tarjeta CineCampa? (s/n): n

Aquí tiene sus entradas. Gracias por su compra.
Entradas individuales          4
Precio por entrada individual  8.00 €
Total                         32.00 €
Descuento                      0.00 €
A pagar                       32.00 €
```

Ejemplo 2:

```
Venta de entradas CineCampa
Número de entradas: 4
Día de la semana: viernes
¿Tiene tarjeta CineCampa? (s/n): s

Aquí tiene sus entradas. Gracias por su compra.
Entradas individuales          4
Precio por entrada individual  8.00 €
Total                         32.00 €
Descuento                      3.20 €
A pagar                       28.80 €
```

Ejemplo 3:

```
Venta de entradas CineCampa
Número de entradas: 4
Día de la semana: jueves
¿Tiene tarjeta CineCampa? (s/n): n

Aquí tiene sus entradas. Gracias por su compra.
Entradas de parejas            2
Precio por entrada de pareja  11.00 €
Total                         22.00 €
Descuento                      0.00 €
A pagar                       22.00 €
```

Ejemplo 4:

```
Venta de entradas CineCampa
Número de entradas: 5
Día de la semana: jueves
¿Tiene tarjeta CineCampa? (s/n): s

Aquí tiene sus entradas. Gracias por su compra.
Entradas parejas               2
```

```
Precio por entrada de pareja  11.00 €
Entradas individuales          1
Precio por entrada individual  8.00 €
Total                         30.00 €
Descuento                      3.00 €
A pagar                       27.00 €
```

Ejemplo 5:

```
Venta de entradas CineCampa
Número de entradas: 5
Día de la semana: miercoles
¿Tiene tarjeta CineCampa? (s/n): s

Aquí tiene sus entradas. Gracias por su compra.
Entradas individuales          5
Precio por entrada individual  5.00 €
Total                         25.00 €
Descuento                      2.50 €
A pagar                       22.50 €
```



Ejercicio 27

Una pastelería nos ha pedido realizar un programa que haga presupuestos de tartas. El programa preguntará primero de qué sabor quiere el usuario la tarta: manzana, fresa o chocolate. La tarta de manzana vale 18 euros y la de fresa 16. En caso de seleccionar la tarta de chocolate, el programa debe preguntar además si el chocolate es negro o blanco; la primera opción vale 14 euros y la segunda 15. Por último se pregunta si se añade nata y si se personaliza con un nombre; la nata suma 2.50 y la escritura del nombre 2.75.

Ejemplo 1:

```
Elija un sabor (manzana, fresa o chocolate): chocolate
¿Qué tipo de chocolate quiere? (negro o blanco): negro
¿Quiere nata? (si o no): si
¿Quiere ponerle un nombre? (si o no): no
Tarta de chocolate negro: 14,00 €
Con nata: 2,50 €
Total: 16,50 €
```

Ejemplo 2:

```
Elija un sabor (manzana, fresa o chocolate): manzana
¿Quiere nata? (si o no): no
```

```
¿Quiere ponerle un nombre? (si o no): si
Tarta de manzana: 18,00 €
Con nombre: 2,75 €
Total: 20,75 €
```

Ejemplo 3:

```
Elija un sabor (manzana, fresa o chocolate): fresa
¿Quiere nata? (si o no): si
¿Quiere ponerle un nombre? (si o no): si
Tarta de fresa: 16,00 €
Con nata: 2,50 €
Con nombre: 2,75 €
Total: 21,25 €
```



Ejercicio 28

Implementa el juego **piedra, papel y tijera**. Primero, el usuario 1 introduce su jugada y luego el usuario 2. Si alguno de los usuarios introduce una opción incorrecta, el programa deberá mostrar un mensaje de error.

Ejemplo 1:

```
Turno del jugador 1 (introduzca piedra, papel o tijera): papel
Turno del jugador 2 (introduzca piedra, papel o tijera): papel
Empate
```

Ejemplo 2:

```
Turno del jugador 1 (introduzca piedra, papel o tijera): papel
Turno del jugador 2 (introduzca piedra, papel o tijera): tijera
Gana el jugador 2
```

Ejemplo 3:

```
Turno del jugador 1 (introduzca piedra, papel o tijera): piedra
Turno del jugador 2 (introduzca piedra, papel o tijera): tijera
Gana el jugador 1
```



Ejercicio 29

Realiza un programa que calcule el precio de un desayuno. El programa preguntará primero qué ha tomado el usuario de comer: palmera, donut o pitufo. La palmera vale 1.40 € y el donut 1 €. En caso de tomar pitufo, el programa debe preguntar además si era con aceite o con tortilla; el primero vale 1’20 € y el segundo 1’60 €. Por último se pregunta por la bebida: zumo o café a 1’50 y 1’20 respectivamente.

Ejemplo 1:

```
¿Qué ha tomado de comer? (palmera, donut o pitufo): palmera
¿Qué ha tomado de beber? (zumo o café): café
Palmera: 1,40 €
Café: 1,20 €
Total desayuno: 2,60 €
```

Ejemplo 2:

```
¿Qué ha tomado de comer? (palmera, donut o pitufo): pitufo
¿Con qué se ha tomado el pitufo? (aceite o tortilla): tortilla
¿Qué ha tomado de beber? (zumo o café): zumo
Pitufo con tortilla: 1,60 €
Zumo: 1,50 €
Total desayuno: 3,10 €
```

5. Bucles

Los bucles se utilizan para repetir un conjunto de sentencias. Por ejemplo, imagina que es necesario introducir la notas de 40 alumnos con el fin de calcular la media, la nota máxima y la nota mínima. Podríamos escribir 40 veces la instrucción que pide un dato por teclado `System.console().readLine()` y convertir cada uno de esos datos a un número con decimales con 40 instrucciones `Double.parseDouble()`, no parece algo muy eficiente. Es mucho más práctico meter dentro de un bucle aquellas sentencias que queremos que se repitan.

Normalmente existe una condición de salida, que hace que el flujo del programa abandone el bucle y continúe justo en la siguiente sentencia. Si no existe condición de salida o si esta condición no se cumple nunca, se produciría lo que se llama un bucle infinito y el programa no terminaría nunca.

5.1 Bucle `for`

Se suele utilizar cuando se conoce previamente el número exacto de iteraciones (repeticiones) que se van a realizar. La sintaxis es la siguiente:

```
for (expresion1 ; expresion2 ; expresion3) {

  sentencias

}
```

Justo al principio se ejecuta `expresion1` y normalmente se usa para inicializar una variable. El bucle se repite mientras se cumple `expresion2` y en cada iteración del bucle se ejecuta `expresion3`, que suele ser el incremento o decremento de una variable. Con un ejemplo se verá mucho más claro.

```
/**
 * Bucle for
 *
 * @author Luis José Sánchez
 */
public class EjemploFor {
  public static void main(String[] args) {
    for (int i = 1; i < 11; i++) {
      System.out.println(i);
    }
  }
}
```

En este ejemplo, `int i = 1` se ejecuta solo una vez, antes que cualquier otra cosa; como ves, esta expresión se utiliza para inicializar la variable `i` a 1. Mientra se cumpla la condición `i < 11` el contenido del bucle, o sea, `System.out.println(i);` se va a ejecutar. En cada iteración del bucle, `i++` hace que la variable `i` se incremente en 1. El resultado del ejemplo es la impresión en pantalla de los números del 1 al 10.

Intenta seguir mentalmente el flujo del programa. Experimenta inicializando la variable `i` con otros valores, cambia la condición con `>` o `<=` y observa lo que sucede. Prueba también a cambiar el incremento de la variable `i`, por ejemplo con `i = i + 2`.

5.2 Bucle `while`

El bucle `while` se utiliza para repetir un conjunto de sentencias siempre que se cumpla una determinada condición. Es importante reseñar que la condición se comprueba al comienzo del bucle, por lo que se podría dar el caso de que dicho bucle no se ejecutase nunca. La sintaxis es la siguiente:

```
while (expresion) {

  sentencias

}
```

Las sentencias se ejecutan una y otra vez mientras `expresion` sea verdadera. El siguiente ejemplo produce la misma salida que el ejemplo anterior, muestra cómo cambian los valores de `i` del 1 al 10.

```
/**
 * Bucle while
 *
 * @author Luis José Sánchez
 */
public class EjemploWhile {
  public static void main(String[] args) {
    int i = 1;

    while (i < 11) {
      System.out.println(i);
      i++;
    }
  }
}
```

En el siguiente ejemplo se cuentan y se suman los números que se van introduciendo por teclado. Para indicarle al programa que debe dejar de pedir números, el usuario debe introducir un número negativo; esa será la condición de salida del bucle. Observa que el bucle se repite mientras el número introducido sea mayor o igual que cero.

```
/**
 * Bucle while que termina cuando se introduce por teclado un
 * número negativo.
 *
 * @author Luis José Sánchez
 */
public class CuentaPositivos {
  public static void main(String[] args) {
    System.out.println("Por favor, vaya introduciendo números y pulsando INTRO.");
    System.out.println("Para terminar, introduzca un número negativo.");

    int numeroIntroducido = 0;
    int cuentaNumeros = 0;
    int suma = 0;

    while (numeroIntroducido >= 0) {
      numeroIntroducido = Integer.parseInt(System.console().readLine());
      cuentaNumeros++; // Incrementa en uno la variable
      suma += numeroIntroducido; // Equivale a suma = suma + NumeroIntroducido
    }

    System.out.println("Has introducido " + (cuentaNumeros - 1) + " números positivos.");
    System.out.println("La suma total de ellos es " + (suma - numeroIntroducido));
  }
}
```

5.3 Bucle `do-while`

El bucle `do-while` funciona de la misma manera que el bucle `while`, con la salvedad de que `expresion` se evalúa al final de la iteración. Las sentencias que encierran el bucle `do-while`, por tanto, se ejecutan como mínimo una vez. La sintaxis es la siguiente:

```
do {

  sentencias

} while (expresion)
```

El siguiente ejemplo es el equivalente `do-while` a los dos ejemplos anteriores que cuentan del 1 al 10.

```
/**
 * Bucle do-while
 *
 * @author Luis José Sánchez
 */
public class EjemploDoWhile {
  public static void main(String[] args) {
    int i = 1;

    do {
      System.out.println(i);
      i++;
    } while (i < 11);
  }
}
```

Veamos otro ejemplo. En este caso se van a ir leyendo números de teclado mientras el número introducido sea par; el programa parará, por tanto, cuando se introduzca un número impar.

```
/**
 * Bucle do-while que termina cuando se introduce por teclado un
 * número impar.
 *
 * @author Luis José Sánchez
 */
public class TerminaCuandoEsImpar {
  public static void main(String[] args) {
    int numero;

    do {
      System.out.print("Dime un número: ");
      numero = Integer.parseInt(System.console().readLine());

      if (numero % 2 == 0) {// comprueba si el número introducido es par
        System.out.println("Qué bonito es el " + numero);
      } else {
        System.out.println("No me gustan los números impares, adiós.");
      }
    } while (numero % 2 == 0);
  }
}
```

Te invito a que realices una modificación sobre este último ejemplo. Después de pedir un número, haz que el programa diga `¿Quiere continuar? (s/n)`. Si el usuario introduce una `s` o una `S`, el programa deberá continuar pidiendo números.

Cualquier bucle que un programador quiera realizar en Java lo puede implementar con `for`, `while` o `do-while`; por tanto, con una sola de estas tres opciones tendría suficiente. No obstante, según el programa en cuestión, una u otra posibilidad se adapta mejor que las otras y resulta más elegante. Con la experiencia, te irás dando cuenta cuándo es mejor utilizar cada una.

5.4 Ejercicios



Ejercicio 1

Muestra los números múltiplos de 5 de 0 a 100 utilizando un bucle `for`.



Ejercicio 2

Muestra los números múltiplos de 5 de 0 a 100 utilizando un bucle `while`.



Ejercicio 3

Muestra los números múltiplos de 5 de 0 a 100 utilizando un bucle `do-while`.



Ejercicio 4

Muestra los números del 320 al 160, contando de 20 en 20 hacia atrás utilizando un bucle `for`.



Ejercicio 5

Muestra los números del 320 al 160, contando de 20 en 20 hacia atrás utilizando un bucle `while`.



Ejercicio 6

Muestra los números del 320 al 160, contando de 20 en 20 utilizando un bucle `do-while`.



Ejercicio 7

Realiza el control de acceso a una caja fuerte. La combinación será un número de 4 cifras. El programa nos pedirá la combinación para abrirla. Si no acertamos, se nos mostrará el mensaje "Lo siento, esa no es la combinación" y si acertamos se nos dirá "La caja fuerte se ha abierto satisfactoriamente". Tendremos cuatro oportunidades para abrir la caja fuerte.



Ejercicio 8

Muestra la tabla de multiplicar de un número introducido por teclado.



Ejercicio 9

Realiza un programa que nos diga cuántos dígitos tiene un número introducido por teclado. Este ejercicio es equivalente a otro realizado anteriormente, con la salvedad de que el anterior estaba limitado a números de 5 dígitos como máximo. En esta ocasión, hay que realizar el ejercicio utilizando bucles; de esta manera, la única limitación en el número de dígitos la establece el tipo de dato que se utilice (`int` o `long`).



Ejercicio 10

Escribe un programa que calcule la media de un conjunto de números positivos introducidos por teclado. A priori, el programa no sabe cuántos números se introducirán. El usuario indicará que ha terminado de introducir los datos cuando meta un número negativo.



Ejercicio 11

Escribe un programa que muestre en tres columnas, el cuadrado y el cubo de los 5 primeros números enteros a partir de uno que se introduce por teclado.



Ejercicio 12

Escribe un programa que muestre los n primeros términos de la serie de Fibonacci. El primer término de la serie de Fibonacci es 0, el segundo es 1 y el resto se calcula sumando los dos anteriores, por lo que tendríamos que los términos son 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144… El número n se debe introducir por teclado.



Ejercicio 13

Escribe un programa que lea una lista de diez números y determine cuántos son positivos, y cuántos son negativos.



Ejercicio 14

Escribe un programa que pida una base y un exponente (entero positivo) y que calcule la potencia.



Ejercicio 15

Escribe un programa que dados dos números, uno real (base) y un entero positivo (exponente), saque por pantalla todas las potencias con base el numero dado y exponentes entre uno y el exponente introducido. No se deben utilizar funciones de exponenciación. Por ejemplo, si introducimos el 2 y el 5, se deberán mostrar 2^1, 2^2, 2^3, 2^4 y 2^5.



Ejercicio 16

Escribe un programa que diga si un número introducido por teclado es o no primo. Un número primo es aquel que sólo es divisible entre él mismo y la unidad.



Ejercicio 17

Realiza un programa que sume los 100 números siguientes a un número entero y positivo introducido por teclado. Se debe comprobar que el dato introducido es correcto (que es un número positivo).



Ejercicio 18

Escribe un programa que obtenga los números enteros comprendidos entre dos números introducidos por teclado y validados como distintos, el programa debe empezar por el menor de los enteros introducidos e ir incrementando de 7 en 7.



Ejercicio 19

Realiza un programa que pinte una pirámide por pantalla. La altura se debe pedir por teclado. El carácter con el que se pinta la pirámide también se debe pedir por teclado.



Ejercicio 20

Igual que el ejercicio anterior pero esta vez se debe pintar una pirámide hueca.



Ejercicio 21

Realiza un programa que vaya pidiendo números hasta que se introduzca un numero negativo y nos diga cuantos números se han introducido, la media de los impares y el mayor de los pares. El número negativo sólo se utiliza para indicar el final de la introducción de datos pero no se incluye en el cómputo.



Ejercicio 22

Muestra por pantalla todos los números primos entre 2 y 100, ambos incluidos.



Ejercicio 23

Escribe un programa que permita ir introduciendo una serie indeterminada de números mientras su suma no supere el valor 10000. Cuando esto último ocurra, se debe mostrar el total acumulado, el contador de los números introducidos y la media.



Ejercicio 24

Escribe un programa que lea un número `n` e imprima una pirámide de números con `n` filas como en la siguiente figura:

```
   1
  121
 12321
1234321
```



Ejercicio 25

Realiza un programa que pida un número por teclado y que luego muestre ese número al revés.



Ejercicio 26

Realiza un programa que pida primero un número y a continuación un dígito. El programa nos debe dar la posición (o posiciones) contando de izquierda a derecha que ocupa ese dígito en el número introducido.



Ejercicio 27

Escribe un programa que muestre, cuente y sume los múltiplos de 3 que hay entre 1 y un número leído por teclado.



Ejercicio 28

Escribe un programa que calcule el factorial de un número entero leído por teclado.

Ejemplo:

```
Por favor, introduzca un número entero: 6
6! = 720
```



Ejercicio 29

Escribe un programa que muestre por pantalla todos los números enteros positivos menores a uno leído por teclado que no sean divisibles entre otro también leído de igual forma.



Ejercicio 30

Realiza una programa que calcule las horas transcurridas entre dos horas de dos días de la semana. No se tendrán en cuenta los minutos ni los segundos. El día de la semana se puede pedir como un número (del 1 al 7) o como una cadena (de “lunes” a “domingo”). Se debe comprobar que el usuario introduce los datos correctamente y que el segundo día es posterior al primero.

Ejemplo:

```
Por favor, introduzca la primera hora.
Día: lunes
Hora: 18
Por favor, introduzca la segunda hora.
Día: martes
Hora: 20
Entre las 18:00h del lunes y las 20:00h del martes hay 26 hora/s.
```



Ejercicio 31

Realiza un programa que pinte la letra **L** por pantalla hecha con asteriscos. El programa pedirá la altura. El palo horizontal de la **L** tendrá una longitud de la mitad (división entera entre 2) de la altura más uno.

Ejemplo:

```
Introduzca la altura de la L: 5

*
*
*
*
* * *
```



Ejercicio 32

Escribe un programa que, dado un número entero positivo, diga cuáles son y cuánto suman los dígitos pares. Los dígitos pares se deben mostrar en orden, de izquierda a derecha. Usa `long` en lugar de `int` donde sea necesario para admitir números largos.

Ejemplo 1:

```
Por favor, introduzca un número entero positivo: 94026782
Dígitos pares: 4 0 2 6 8 2
Suma de los dígitos pares: 22
```

Ejemplo 2:

```
Por favor, introduzca un número entero positivo: 31779
Dígitos pares:
Suma de los dígitos pares: 0
```

Ejemplo 3:

```
Por favor, introduzca un número entero positivo: 2404
Dígitos pares: 2 4 0 4
Suma de los dígitos pares: 10
```



Ejercicio 33

Realiza un programa que pinte la letra **U** por pantalla hecha con asteriscos. El programa pedirá la altura. Fíjate que el programa inserta un espacio y pinta dos asteriscos menos en la base para simular la curvatura de las esquinas inferiores.

Ejemplo 1:

```
Introduzca la altura de la U: 5

 *       *
 *       *
 *       *
 *       *
   * * *
```

Ejemplo 2:

```
Introduzca la altura de la U: 4

 *     *
 *     *
 *     *
   * *
```



Ejercicio 34

Escribe un programa que pida dos números por teclado y que luego mezcle en dos números diferentes los dígitos pares y los impares. Se van comprobando los dígitos de la siguiente manera: primer dígito del primer número, primer dígito del segundo número, segundo dígito del primer número, segundo dígito del segundo número, tercer dígito del primer número… Para facilitar el ejercicio, podemos suponer que el usuario introducirá dos números de la misma longitud y que siempre habrá al menos un dígito par y uno impar. Usa `long` en lugar de `int` donde sea necesario para admitir números largos.

Ejemplo 1:

```
Por favor, introduzca un número: 9402
Introduzca otro número: 6782
El número formado por los dígitos pares es 640822
El número formado por los dígitos impares es 97
```

Ejemplo 2:

```
Por favor, introduzca un número: 137
Introduzca otro número: 909
El número formado por los dígitos pares es 0
El número formado por los dígitos impares es 19379
```



Ejercicio 35

Realiza un programa que pinte una X hecha de asteriscos. El programa debe pedir la altura. Se debe comprobar que la altura sea un número impar mayor o igual a 3, en caso contrario se debe mostrar un mensaje de error.

Ejemplo:

```
Por favor, introduzca la altura de la X: 5

*   *
 * *
  *
 * *
*   *
```



Ejercicio 36

Escribe un programa que diga si un número introducido por teclado es o no capicúa. Los números capicúa se leen igual hacia delante y hacia atrás. El programa debe aceptar números de cualquier longitud siempre que lo permita el tipo, en caso contrario el ejercicio no se dará por bueno. Se recomienda usar `long` en lugar de `int` ya que el primero admite números más largos.

Ejemplo 1:

```
Por favor, introduzca un número entero positivo: 678
El 678 no es capicúa.
```

Ejemplo 2:

```
Por favor, introduzca un número entero positivo: 2019102
El 2019102 es capicúa.
```



Ejercicio 37

Realiza un conversor del sistema decimal al sistema de "palotes".

Ejemplo:

```
Por favor, introduzca un número entero positivo: 47021
El 47021 en decimal es el | | | | - | | | | | | | - - | | - | en el sistema de palotes.
```



Ejercicio 38

Realiza un programa que pinte un reloj de arena relleno hecho de asteriscos. El programa debe pedir la altura. Se debe comprobar que la altura sea un número impar mayor o igual a 3, en caso contrario se debe mostrar un mensaje de error.

Ejemplo:

```
Por favor, introduzca la altura del reloj de arena: 5

*****
 ***
  *
 ***
*****
```



Ejercicio 39

Escribe un programa que pida un número entero positivo por teclado y que muestre a continuación los números desde el 1 al número introducido junto con su factorial.

Ejemplo:

```
Por favor, introduzca un número entero positivo: 7

1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
7! = 5040
```



Ejercicio 40

Realiza un programa que pinte por pantalla un rombo hueco hecho con asteriscos. El programa debe pedir la altura. Se debe comprobar que la altura sea un número impar mayor o igual a 3, en caso contrario se debe mostrar un mensaje de error.

Ejemplo:

```
Por favor, introduzca la altura del rombo: 5

  *
 * *
*   *
 * *
  *
```



Ejercicio 41

Escribe un programa que diga cuántos dígitos pares y cuántos dígitos impares hay dentro de un número. Se recomienda usar `long` en lugar de `int` ya que el primero admite números más largos.

Ejemplo 1:

```
Por favor, introduzca un número entero positivo: 406783
El 406783 contiene 4 dígitos pares y 2 dígitos impares.
```

Ejemplo 2:

```
Por favor, introduzca un número entero positivo: 3177840
El 3177840 contiene 3 dígitos pares y 4 dígitos impares.
```



Ejercicio 42

Escribe un programa que pida un número entero positivo por teclado y que muestre a continuación los 5 números consecutivos a partir del número introducido. Al lado de cada número se debe indicar si se trata de un primo o no.

Ejemplo:

```
Por favor, introduzca un número entero positivo: 17
17 es primo
18 no es primo
19 es primo
20 no es primo
21 no es primo
```



Ejercicio 43

Escribe un programa que permita partir un número introducido por teclado en dos partes. Las posiciones se cuentan de izquierda a derecha empezando por el 1. Suponemos que el usuario introduce correctamente los datos, es decir, el número introducido tiene dos dígitos como mínimo y la posición en la que se parte el número está entre 2 y la longitud del número. No se permite en este ejercicio el uso de funciones de manejo de `String` (por ej. para extraer subcadenas dentro de una cadena).

Ejemplo:

```
Por favor, introduzca un número entero positivo: 406783
Introduzca la posición a partir de la cual quiere partir el número: 5
Los números partidos son el 4067 y el 83.
```



Ejercicio 44

Escribe un programa que sea capaz de insertar un dígito dentro de un número indicando la posición. El nuevo dígito se colocará en la posición indicada y el resto de dígitos se desplazará hacia la derecha. Las posiciones se cuentan de izquierda a derecha empezando por el 1. Suponemos que el usuario introduce correctamente los datos. Se recomienda usar `long` en lugar de `int` ya que el primero admite números más largos.

Ejemplo:

```
Por favor, introduzca un número entero positivo: 406783
Introduzca la posición donde quiere insertar: 3
Introduzca el dígito que quiere insertar: 5
El número resultante es 4056783.
```



Ejercicio 45

Escribe un programa que cambie un dígito dentro de un número dando la posición y el valor nuevo. Las posiciones se cuentan de izquierda a derecha empezando por el 1. Se recomienda usar `long` en lugar de `int` ya que el primero admite números más largos. Suponemos que el usuario introduce correctamente los datos.

Ejemplo:

```
Por favor, introduzca un número entero positivo: 406783
Introduzca la posición dentro del número: 3
Introduzca el nuevo dígito: 1
El número resultante es 401783
```



Ejercicio 46

Realiza un programa que pinte por pantalla un rectángulo hueco hecho con asteriscos. Se debe pedir al usuario la anchura y la altura. Hay que comprobar que tanto la anchura como la altura sean mayores o iguales que 2, en caso contrario se debe mostrar un mensaje de error.

Ejemplo 1:

```
Por favor, introduzca la anchura del rectángulo (como mínimo 2): 4
Ahora introduzca la altura (como mínimo 2): 1
Lo siento, los datos introducidos no son correctos, el valor mínimo para la anchura y la altur\
a es 2.
```

Ejemplo 2:

```
Por favor, introduzca la anchura del rectángulo (como mínimo 2): 6
Ahora introduzca la altura (como mínimo 2): 4

* * * * * *
*         *
*         *
* * * * * *
```



Ejercicio 47

Con motivo de la celebración del día de la mujer, el 8 de marzo, nos han encargado realizar un programa que pinte un 8 por pantalla usando la letra M. Se pide al usuario la altura, que debe ser un número entero impar mayor o igual que 5. Si el número introducido no es correcto, el programa deberá mostrar un mensaje de error. A continuación se muestran algunos ejemplos. La anchura de la figura siempre será de 6 caracteres.

Ejemplo 1:

```
Por favor, introduzca la altura (número impar mayor o igual a 5): 8
La altura introducida no es correcta
```

Ejemplo 2:

```
Por favor, introduzca la altura (número impar mayor o igual a 5): 3
La altura introducida no es correcta
```

Ejemplo 3:

```
Por favor, introduzca la altura (número impar mayor o igual a 5): 5
MMMMMM
M    M
MMMMMM
M    M
MMMMMM
```

Ejemplo 4:

```
Por favor, introduzca la altura (número impar mayor o igual a 5): 9
MMMMMM
M    M
M    M
M    M
MMMMMM
M    M
M    M
M    M
MMMMMM
```



Ejercicio 48

Realiza un programa que diga los dígitos que aparecen y los que no aparecen en un número entero introducido por teclado. El orden es el que se muestra en los ejemplos. Utiliza el tipo long para que el usuario pueda introducir números largos.

Ejemplo 1:

```
Introduzca un número entero: 67706
Dígitos que aparecen en el número: 0 6 7
Dígitos que no aparecen: 1 2 3 4 5 8 9
```

Ejemplo 2:

```
Introduzca un número entero: 555
Dígitos que aparecen en el número: 5
Dígitos que no aparecen: 1 2 3 4 6 7 8 9
```

Ejemplo 3:

```
Introduzca un número entero: 9876543210
Dígitos que aparecen en el número: 0 1 2 3 4 5 6 7 8 9
Dígitos que no aparecen:
```

Ejemplo 4:

```
Introduzca un número entero: 13247721
Dígitos que aparecen en el número: 1 2 3 4 7
Dígitos que no aparecen: 0 5 6 8 9
```



Ejercicio 49

Realiza un programa que calcule el máximo, el mínimo y la media de una serie de números enteros positivos introducidos por teclado. El programa terminará cuando el usuario introduzca un número primo. Este último número no se tendrá en cuenta en los cálculos. El programa debe indicar también cuántos números ha introducido el usuario (sin contar el primo que sirve para salir).

Ejemplo:

```
Por favor, vaya introduciendo números enteros positivos. Para terminar, introduzca un número p\
rimo:
6
8
15
12
23
Ha introducido 4 números no primos.
Máximo: 15
Mínimo: 6
Media: 10.25
```



Ejercicio 50

Una empresa de cartelería nos ha encargado un programa para realizar uno de sus diseños. Debido a los acontecimientos que han tenido lugar en Cataluña durante el 2018, han recibido muchos pedidos del cartel que muestra el número 155. Realiza un programa que pinte el número 155 mediante asteriscos. Al usuario se le pedirán dos datos, la altura del cartel y el número de espacios que habrá entre los números. La altura mínima es 5. La anchura de los números siempre es la misma. La parte superior de los cincos también es siempre igual. La parte inferior del 5 sí que varía en función de la altura.

Ejemplo 1:

Introduzca la altura (5 como mínimo): 5 Introduzca el número de espacios entre los números (1 como mínimo): 2

```
*  ****  ****
*  *     *
*  ****  ****
*     *     *
*  ****  ****
```

Ejemplo 2:

Introduzca la altura (5 como mínimo): 7 Introduzca el número de espacios entre los números (1 como mínimo): 3

```
*   ****   ****
*   *      *
*   ****   ****
*      *      *
*      *      *
*      *      *
*   ****   ****
```

Ejemplo 3:
Introduzca la altura (5 como mínimo): 6 Introduzca el número de espacios entre los números (1 como mínimo): 1

```
* **** ****
* *    *
* **** ****
*    *    *
*    *    *
* **** ****
```



Ejercicio 51

El gusano numérico se come los dígitos con forma de rosquilla, o sea, el 0 y el 8 (todos los que encuentre). Realiza un programa que muestre un número antes y después de haber sido comido por el gusano. Si el animalito no se ha comido ningún dígito, el programa debe indicarlo.

Ejemplo 1:

```
Introduzca un número entero (mayor que cero): 51803458
Después de haber sido comido por el gusano numérico se queda en 51345
```

Ejemplo 2:

```
Introduzca un número entero (mayor que cero): 29614
El gusano numérico no se ha comido ningún dígito.
```



Ejercicio 52

Realiza un programa que sea capaz de desplazar todos los dígitos de un número de derecha a izquierda una posición. El dígito de más a la izquierda, pasaría a dar la vuelta y se colocaría a la derecha. Si el número tiene un solo dígito, se queda igual.

Ejemplo 1:

```
Introduzca un número: 609831
El número resultado es 98316
```

Ejemplo 2:

```
Introduzca un número: 78201345
El número resultado es 82013457
```

Ejemplo 3:

```
Introduzca un número: 24
El número resultado es 42
```

Ejemplo 4:

```
Introduzca un número: 8
El número resultado es 8
```



Ejercicio 53

Realiza un programa que pinte un triángulo relleno tal como se muestra en los ejemplos. El usuario debe introducir la altura de la figura.

Ejemplo 1:

```
Introduzca la altura de la figura: 8
********
*******
******
*****
****
***
**
*
```

Ejemplo 2:

```
Introduzca la altura de la figura: 5
*****
****
***
**
*
```



Ejercicio 54

Realiza un programa que pinte un triángulo hueco tal como se muestra en los ejemplos. El usuario debe introducir la altura de la figura.

Ejemplo 1:

```
Introduzca la altura de la figura: 8
********
*     *
*    *
*   *
*  *
* *
**
*
```

Ejemplo 2:

```
Introduzca la altura de la figura: 5
*****
*  *
* *
**
*
```



Ejercicio 55

Realiza un programa que sea capaz de desplazar todos los dígitos de un número de izquierda a derecha una posición. El dígito de más a la derecha, pasaría a dar la vuelta y se colocaría a la izquierda. Si el número tiene un solo dígito, se queda igual.

Ejemplo 1:

```
Introduzca un número: 609831
El número resultado es 160983
```

Ejemplo 2:

```
Introduzca un número: 78201345
El número resultado es 57820134
```

Ejemplo 3:

```
Introduzca un número: 24
El número resultado es 42
```

Ejemplo 4:

```
Introduzca un número: 8
El número resultado es 8
```



Ejercicio 56

Realiza un programa que pinte un triángulo relleno tal como se muestra en los ejemplos. El usuario debe introducir la altura de la figura.

Ejemplo 1:

```
Introduzca la altura de la figura: 8
********
 *******
  ******
   *****
    ****
     ***
      **
       *
```

Ejemplo 2:

```
Introduzca la altura de la figura: 5
*****
 ****
  ***
   **
    *
```



Ejercicio 57

Realiza un programa que pinte un triángulo hueco tal como se muestra en los ejemplos. El usuario debe introducir la altura de la figura.

Ejemplo 1:

```
Introduzca la altura de la figura: 8
********
 *     *
  *    *
   *   *
    *  *
     * *
      **
       *
```

Ejemplo 2:

```
Introduzca la altura de la figura: 5
*****
 *  *
  * *
   **
    *
```



Ejercicio 58

Realiza un programa que calcule la media de los dígitos que contiene un número entero introducido por teclado.

Ejemplo 1:

```
Introduzca un número: 609831
La media de sus dígitos es 4.5
```

Ejemplo 2:

```
Introduzca un número: 78201345
La media de sus dígitos es 3.75
```

Ejemplo 3:

```
Introduzca un número: 24
La media de sus dígitos es 3.0
```

Ejemplo 4:

```
Introduzca un número: 8
La media de sus dígitos es 8.0
```



Ejercicio 59

Escribe un programa que pinte por pantalla un árbol de navidad. El usuario debe introducir la altura. En esa altura va incluida la estrella y el tronco. Suponemos que el usuario introduce una altura mayor o igual a 4.

Ejemplo 1:

```
Por favor, introduzca la altura del árbol: 7
    *
    ^
   ^ ^
  ^   ^
 ^     ^
^^^^^^^^^
    Y
```

Ejemplo 2:

```
Por favor, introduzca la altura del árbol: 4
 *
 ^
^^^
 Y
```

Ejemplo 3:

```
Por favor, introduzca la altura del árbol: 10
       *
       ^
      ^ ^
     ^   ^
    ^     ^
   ^       ^
  ^         ^
 ^           ^
^^^^^^^^^^^^^^^
       Y
```



Ejercicio 60

Escribe un programa que pinte por pantalla un par de calcetines, de los que se ponen al lado del árbol de Navidad para que Papá Noel deje sus regalos. El usuario debe introducir la altura. Suponemos que el usuario introduce una altura mayor o igual a 4. Observa que la talla de los calcetines y la distancia que hay entre ellos (dos espacios) no cambia, lo único que varía es la altura.

Ejemplo 1:

```
Introduzca la altura de los calcetines: 7

***     ***
***     ***
***     ***
***     ***
***     ***
******  ******
******  ******
```

Ejemplo 2:

```
Introduzca la altura de los calcetines: 4

***     ***
***     ***
******  ******
******  ******
```

Ejemplo 3:

```
Introduzca la altura de los calcetines: 9

***     ***
***     ***
***     ***
***     ***
***     ***
***     ***
***     ***
******  ******
******  ******
```



Ejercicio 61

Escribe un programa que pinte por pantalla la letra V. El ancho del palo de la V es siempre de 3 asteriscos. El usuario debe introducir la altura. La altura mínima es de 3 pisos. Si el usuario introduce una altura menor, el programa debe mostrar un mensaje de error.

Ejemplo 1:

```
Introduzca la altura de la V (un número mayor o igual a 3): 7

***            ***
 ***          ***
  ***        ***
   ***      ***
    ***    ***
     ***  ***
      ******
```

Ejemplo 2:

```
Introduzca la altura de la V (un número mayor o igual a 3): 4

***      ***
 ***    ***
  ***  ***
   ******
```

Ejemplo 3:

```
Introduzca la altura de la V (un número mayor o igual a 3): 9

***                ***
 ***              ***
  ***            ***
   ***          ***
    ***        ***
     ***      ***
      ***    ***
       ***  ***
        ******
```

Ejemplo 4:

```
Introduzca la altura de la V (un número mayor o igual a 3): 2
La altura debe ser mayor o igual a 3.
```



Ejercicio 62

Según cierta cultura oriental, los números de la suerte son el 3, el 7, el 8 y el 9. Los números de la mala suerte son el resto: el 0, el 1, el 2, el 4, el 5 y el 6. Un número es **afortunado** si contiene más números de la suerte que de la mala suerte. Realiza un programa que diga si un número introducido por el usuario es afortunado o no.

Ejemplo 1:

```
Introduzca un número: 772
El 772 es un número afortunado.
```

Ejemplo 2:

```
Introduzca un número: 7720
El 7720 no es un número afortunado.
```

Ejemplo 3:

```
Introduzca un número: 43081
El 43081 no es un número afortunado.
```

Ejemplo 4:

```
Introduzca un número: 888
El 888 es un número afortunado.
```

Ejemplo 5:

```
Introduzca un número: 1234
El 1234 no es un número afortunado.
```

Ejemplo 6:

```
Introduzca un número: 6789
El 6789 es un número afortunado.
```



Ejercicio 63

Realiza un programa que pinte dos pirámides rellenas hechas con asteriscos, una al lado de la otra y separadas por un espacio en su base.

Ejemplo 1:

```
Introduzca la altura de la primera pirámide: 7
Introduzca la altura de la segunda pirámide: 3

      *
     ***
    *****
   *******
  *********     *
 ***********   ***
************* *****
```

Ejemplo 2:

```
Introduzca la altura de la primera pirámide: 4
Introduzca la altura de la segunda pirámide: 5

            *
   *       ***
  ***     *****
 *****   *******
******* *********
```



Ejercicio 64

Escribe un programa que pinte por pantalla un **rectángulo hueco** de 6 caracteres de ancho por 3 de alto y, a continuación, un menú que permita **agrandarlo**, **achicarlo** o **cambiar su orientación**. Cada vez que el rectángulo se agranda, se incrementa en 1 tanto su anchura como su altura. Cuando se achica, se decrementa en 1 su anchura y altura. Por último, cuando se cambia la orientación, los valores de anchura y altura se intercambian. El valor mínimo de la altura o la anchura es 2.

Ejemplo:

```
******
*    *
******
1. Agrandarlo
2. Achicarlo
3. Cambiar la orientación
4. Salir
Indique qué quiere hacer con el rectángulo: 2

****
****
1. Agrandarlo
2. Achicarlo
3. Cambiar la orientación
4. Salir
Indique qué quiere hacer con el rectángulo: 2

****
****
1. Agrandarlo
2. Achicarlo
```

```
3. Cambiar la orientación
4. Salir
Indique qué quiere hacer con el rectángulo: 1

*****
*   *
*****
1. Agrandarlo
2. Achicarlo
3. Cambiar la orientación
4. Salir
Indique qué quiere hacer con el rectángulo: 3

***
* *
* *
* *
***
1. Agrandarlo
2. Achicarlo
3. Cambiar la orientación
4. Salir
Indique qué quiere hacer con el rectángulo: 4
```



Ejercicio 65

Escribe un programa que pinte por pantalla la letra A. El usuario debe introducir la altura total y la fila en la que debe aparecer el palito horizontal (contando desde el vértice). La altura mínima es de 3 pisos. La fila donde va el palito horizontal debe ser mayor que 1 y menor que la altura total. Si el usuario introduce algún dato incorrecto, el programa debe mostrar un mensaje de error.

Ejemplo 1:

```
Introduzca la altura de la A (un número mayor o igual a 3): 7
Introduzca la fila del palito horizontal (entre 2 y 6): 5

      *
     * *
    *   *
   *     *
  *********
 *         *
*           *
```

Ejemplo 2:

```
Introduzca la altura de la A (un número mayor o igual a 3): 7
Introduzca la fila del palito horizontal (entre 2 y 6): 6

      *
     * *
    *   *
   *     *
  *       *
 ***********
*           *
```

Ejemplo 3:

```
Introduzca la altura de la A (un número mayor o igual a 3): 7
Introduzca la fila del palito horizontal (entre 2 y 6): 7
La fila introducida no es correcta.
```

Ejemplo 4:

```
Introduzca la altura de la A (un número mayor o igual a 3): 2
La altura introducida no es correcta.
```

Ejemplo 5:

```
Introduzca la altura de la A (un número mayor o igual a 3): 4
Introduzca la fila del palito horizontal (entre 2 y 3): 2

   *
  ***
 *   *
*     *
```

Ejemplo 1:

```
Introduzca la altura de la A (un número mayor o igual a 3): 5
Introduzca la fila del palito horizontal (entre 2 y 4): 4

    *
   * *
  *   *
 *******
*       *
```



Ejercicio 66

La Guardia Civil de Tráfico nos ha encargado un programa que pinte una señal para desviar el tráfico hacia la derecha. La señal es una doble flecha con el vértice apuntando a la derecha. Se pide al usuario la altura de la figura, que debe ser un número impar mayor o igual que 3. La distancia entre cada flecha de asteriscos es siempre de 4 espacios. Si la altura introducida por el usuario no es un número impar mayor o igual que 3, el programa debe mostrar un mensaje de error.

Ejemplo 1:

```
Por favor, introduzca la altura de la figura: 7

*    *
 *    *
  *    *
   *    *
  *    *
 *    *
*    *
```

Ejemplo 2:

```
Por favor, introduzca la altura de la figura: 3

*    *
 *    *
*    *
```

Ejemplo 3:

```
Por favor, introduzca la altura de la figura: 4
La altura no es correcta, debe ser un número impar mayor o igual que 3.
```



Ejercicio 67

Realiza un programa que pinte una escalera que va descendiendo de izquierda a derecha. El programa pedirá el número de escalones y la altura de cada escalón. La anchura de los escalones siempre es la misma: 4 asteriscos.

Ejemplo 1:

```
Introduzca el número de escalones: 4
Introduzca la altura de cada escalón: 2
****
****
********
********
************
************
****************
****************
```

Ejemplo 2:

```
Introduzca el número de escalones: 3
Introduzca la altura de cada escalón: 3
****
****
****
********
********
********
************
************
************
```



Ejercicio 68

Escribe un programa que pida un número por teclado y que luego lo **“disloque”** de tal forma que a cada dígito se le suma 1 si es par y se le resta 1 si es impar. Usa long en lugar de int donde sea necesario para admitir números largos.

Ejemplo 1:

```
Por favor, introduzca un número: 9402
Dislocando el 9402 sale el 8513.
```

Ejemplo 2:

```
Por favor, introduzca un número: 870958422
Dislocando el 870958422 sale el 961849533.
```

Ejemplo 3:

```
Por favor, introduzca un número: 137
Dislocando el 137 sale el 26
```



Ejercicio 69

Realiza un programa que pinte una **pirámide maya**. Por los lados, se trata de una pirámide normal y corriente. Por el centro se van pintando líneas de asteriscos de forma alterna (empezando por la superior): la primera se pinta, la segunda no, la tercera sí, la cuarta no, etc. La terraza de la pirámide siempre tiene 6 asteriscos, por tanto, las líneas centrales que se añaden a la pirámide normal tienen 4 asteriscos. El programa pedirá la altura. Se supone que el usuario introducirá un número entero mayor o igual a 3; no es necesario comprobar los datos de entrada.

Ejemplo 1:

```
Introduzca la altura de la pirámide maya: 5

    ******
   **    **
  **********
 ****    ****
**************
```

Ejemplo 2:

```
Introduzca la altura de la pirámide maya: 8

       ******
      **    **
     **********
    ****    ****
   **************
  ******    ******
 ******************
********    ********
```

6. Números aleatorios

Los números aleatorios se utilizan con frecuencia en programación para emular el comportamiento de algún fenómeno natural, el resultado de un juego de azar o, en general, para generar cualquier valor impredecible *a priori*.

Por ejemplo, se pueden utilizar números aleatorios para generar tiradas de dados de tal forma que, de antemano, no se puede saber el resultado. Antes de tirar un dado no sabemos si saldrá un 3 o un 5; se tratará pues de un número impredecible; lo que sí sabemos es que saldrá un número entre el 1 y el 6, es decir, podemos acotar el rango de los valores que vamos a obtener de forma aleatoria.

6.1 Generación de números aleatorios con y sin decimales

Para generar valores aleatorios utilizaremos `Math.random()`. Esta función genera un número con decimales (de tipo `double`) en el intervalo [0 - 1), es decir, genera un número mayor o igual que 0 y menor que 1.

El siguiente programa genera diez números aleatorios:

```
/**
 *  Generación de números aleatorios.
 *
 *  @author Luis José Sánchez
 */
public class Aleatorio01 {
  public static void main(String[] args) {
    System.out.println("Diez números aleatorios:\n");

    for (int i = 1; i < 11; i++) {
      System.out.println(Math.random());
    }
  }
}
```

La salida del programa puede ser algo como esto:

```
0.30830376099567813
0.8330493363981046
0.2322483682676435
0.3130746053770094
0.34192944433558736
0.9636470440975567
0.3398896383918959
0.79825461825305
0.9868509622870223
0.9967893773101499
```

Te invito a que ejecutes varias veces el programa. Podrás observar que cada vez salen números diferentes, aunque siempre están comprendidos entre 0 y 1 (incluyendo el 0).

Pensarás que no es muy útil generar números aleatorios entre 0 y 1 si lo que queremos es por ejemplo sacar una carta al azar de la baraja española; pero en realidad un número decimal entre 0 y 1 es lo único que nos hace falta para generar cualquier tipo de valor aleatorio siempre y cuando se manipule ese número de la forma adecuada.

Por ejemplo, si queremos generar valores aleatorios entre 0 y 10 (incluyendo el 0 y sin llegar a 10) simplemente tendremos que correr la coma un lugar o, lo que es lo mismo, multiplicar por 10.

```
/**
 *  Generación de números aleatorios.
 *
 *  @author Luis José Sánchez
 */
public class Aleatorio02 {
  public static void main(String[] args) {
    System.out.println("20 números aleatorios entre 0 y 10");
    System.out.println(" sin llegar a 10 (con decimales):");

    for (int i = 1; i <= 20; i++) {
      System.out.println( Math.random()*10 + "  ");
    }
  }
}
```

El programa anterior genera una salida como ésta:

```
0.07637674702636321
1.025787417143682
1.854993461897163
5.690351111720931
3.82310645589797
5.518007662236258
9.23529380254256
3.9201032643833376
5.836554253122096
6.224559064261578
7.652976185871555
4.9922807025365135
7.498156441347868
8.743251697509109
9.727764845406675
2.929766691797686
0.05801413446517634
2.1575652936687284
```

Si queremos generar números enteros en lugar de números con decimales, basta con hacer un casting para convertir los números de tipo `double` en números de tipo `int`. Recuerda que `(int)x` transforma `x` en una variable de tipo entero; si `x` era de tipo `float` o `double`, perdería todos los decimales.

```
/**
 * Generación de números aleatorios.
 *
 * @author Luis José Sánchez
 */
public class Aleatorio03 {
  public static void main(String[] args) {
    System.out.println("20 números aleatorios entre 0 y 9 (sin decimales):");

    for (int i = 1; i <= 20; i++) {
      System.out.print((int)(Math.random()*10) + "  ");
    }

    System.out.println();
  }
}
```

La salida del programa debe ser algo muy parecido a esto:

```
20 números aleatorios entre 0 y 9 (sin decimales):
0 8 0 3 8 8 7 3 2 0 8 2 1 2 9 0 6 4 5 4
```

¿Y si en lugar de generar números enteros entre 0 y 9 queremos generar números entre 1 y 10? Como habrás podido adivinar, simplemente habría que sumar 1 al número generado, de esta forma se "desplazan un paso" los valores generados al azar, de tal forma que el mínimo valor que se produce sería el 0 + 1 = 1 y el máximo sería 9 + 1 = 10.

```java
/**
 *  Generación de números aleatorios.
 *
 *  @author Luis José Sánchez
 */
public class Aleatorio04 {
  public static void main(String[] args) {
    System.out.println("20 números aleatorios entre 1 y 10 (sin decimales):");

    for (int i = 1; i <= 20; i++) {
      System.out.print( (int)(Math.random()*10 + 1) + "  ");
    }

    System.out.println();
  }
}
```

Vamos a ponerlo un poco más difícil. Ahora vamos a generar números enteros entre 50 y 60 ambos incluidos. Primero multiplicamos `Math.random()` por 11, con lo que obtenemos números decimales entre 0 y 10.9999… (sin llegar nunca hasta 11). Luego desplazamos ese intervalo sumando 50 por lo que obtenemos números decimales entre 50 y 60.9999… Por último, quitamos los decimales haciendo casting y *voilà*, ya tenemos números enteros aleatorios entre 50 y 60 ambos incluidos.

```java
/**
 * Generación de números aleatorios.
 *
 * @author Luis José Sánchez
 */
public class Aleatorio05 {
  public static void main(String[] args) {
    System.out.println("20 números aleatorios entre 50 y 60 (sin decimales):");

    for (int i = 1; i <= 20; i++) {
      System.out.print(((int)(Math.random()*11) + 50 ) + "  ");
    }
```

```
    System.out.println();
  }
}
```

6.2 Generación de palabras de forma aleatoria de un conjunto dado

Hemos visto cómo generar números aleatorios con y sin decimales y en diferentes intervalos. Vamos a producir ahora de forma aleatoria una palabra - piedra, papel o tijera - generando primero un número entero entre 0 y 2 y posteriormente haciendo corresponder una palabra a cada número.

```
/**
 *  Generación de números aleatorios.
 *
 *  @author Luis José Sánchez
 */
public class Aleatorio06 {
  public static void main(String[] args) {
    System.out.println("Genera al azar piedra, papel o tijera:");

    int mano = (int)(Math.random()*3); // genera un número al azar
                                       // entre 0 y 2 ambos incluidos
    switch(mano) {
      case 0:
        System.out.println("piedra");
        break;
      case 1:
        System.out.println("papel");
        break;
      case 2:
        System.out.println("tijera");
        break;
      default:
    }
  }
}
```

¿Cómo podríamos generar un día de la semana de forma aleatoria? En efecto, primero generamos un número entre 1 y 7 ambos inclusive y luego hacemos corresponder un día de la semana a cada uno de los números.

```
/**
 * Generación de números aleatorios.
 *
 * @author Luis José Sánchez
 */
public class Aleatorio07 {
  public static void main(String[] args) {
    System.out.println("Muestra un día de la semana al azar:");

    int  dia = (int)(Math.random()*7) + 1; // genera un número aleatorio
                                           // entre el 1 y el 7
    switch(dia) {
      case 1:
        System.out.println("lunes");
        break;
      case 2:
        System.out.println("martes");
        break;
      case 3:
        System.out.println("miércoles"); break;
      case 4:
        System.out.println("jueves");
        break;
      case 5:
        System.out.println("viernes");
        break;
      case 6:
        System.out.println("sábado");
        break;
      case 7:
        System.out.println("domingo");
        break;
      default:
    }
  }
}
```

6.3 Ejercicios



Ejercicio 1

Escribe un programa que muestre la tirada de tres dados. Se debe mostrar también la suma total (los puntos que suman entre los tres dados).



Ejercicio 2

Realiza un programa que muestre al azar el nombre de una carta de la baraja francesa. Esta baraja está dividida en cuatro palos: picas, corazones, diamantes y tréboles. Cada palo está formado por 13 cartas, de las cuales 9 cartas son numerales y 4 literales: 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K y A (que sería el 1). Para convertir un número en una cadena de caracteres podemos usar `String.valueOf(n)`.



Ejercicio 3

Igual que el ejercicio anterior pero con la baraja española. Se utilizará la baraja de 40 cartas: 2, 3, 4, 5, 6, 7, sota, caballo, rey y as.



Ejercicio 4

Muestra 20 números enteros aleatorios entre 0 y 10 (ambos incluidos) separados por espacios.



Ejercicio 5

Muestra 50 números enteros aleatorios entre 100 y 199 (ambos incluidos) separados por espacios. Muestra también el máximo, el mínimo y la media de esos números.



Ejercicio 6

Escribe un programa que piense un número al azar entre 0 y 100. El usuario debe adivinarlo y tiene para ello 5 oportunidades. Después de cada intento fallido, el programa dirá cuántas oportunidades quedan y si el número introducido es menor o mayor que el número secreto.



Ejercicio 7

Escribe un programa que muestre tres apuestas de la quiniela en tres columnas para los 14 partidos y el pleno al quince (15 filas).



Ejercicio 8

Modifica el programa anterior para que la probabilidad de que salga un 1 sea de 1/2, la probabilidad de que salga `x` sea de 1/3 y la probabilidad de que salga 2 sea de 1/6. Pista: 1/2 = 3/6 y 1/3 = 2/6.



Ejercicio 9

Realiza un programa que vaya generando números aleatorios pares entre 0 y 100 y que no termine de generar números hasta que no saque el 24. El programa deberá decir al final cuántos números se han generado.



Ejercicio 10

Realiza un programa que pinte por pantalla diez líneas formadas por caracteres. El carácter con el que se pinta cada línea se elige de forma aleatoria entre uno de los siguientes: *, -, =, ., |, @. Las líneas deben tener una longitud aleatoria entre 1 y 40 caracteres.



Ejercicio 11

Escribe un programa que muestre 20 notas generadas al azar. Las notas deben aparecer de la forma: suspenso, suficiente, bien, notable o sobresaliente. Al final aparecerá el número de suspensos, el número de suficientes, el número de bienes, etc.



Ejercicio 12

Realiza un programa que llene la pantalla de caracteres aleatorios (a lo Matrix) con el código ascii entre el 32 y el 126. Puedes hacer casting con `(char)` para convertir un entero en un carácter.



Ejercicio 13

Escribe un programa que simule la tirada de dos dados. El programa deberá continuar tirando los dados una y otra vez hasta que en alguna tirada los dos dados tengan el mismo valor.



Ejercicio 14

Realiza un programa que haga justo lo contrario a lo que hace el ejercicio 6. El programa intentará adivinar el número que estás pensando - un número entre 0 y 100 - teniendo para ello 5 oportunidades. En cada intento fallido, el programa debe preguntar si el número que estás pensando es mayor o menor que el que te acaba de decir.



Ejercicio 15

Realiza un generador de melodía con las siguientes condiciones:

a) Las notas deben estar generadas al azar. Las 7 notas son **do**, **re**, **mi**, **fa**, **sol**, **la** y **si**.

b) Una melodía está formada por un número aleatorio de notas mayor o igual a 4, menor o igual a 28 y siempre múltiplo de 4 (4, 8, 12…).

c) Cada grupo de 4 notas será un compás y estará separado del siguiente compás mediante la barra vertical "|" (Alt + 1). El final de la melodía se marca con dos barras.

d) La última nota de la melodía debe coincidir con la primera.

Ejemplo 1:

```
do mi fa mi | si do sol fa | fa re si do | sol mi re do ||
```

Ejemplo 2:

```
la re mi sol | fa mi mi si | do la sol fa | fa re si sol | do sol mi re | fa la do la ||
```



Ejercicio 16

Realiza un simulador de máquina tragaperras simplificada que cumpla los siguientes requisitos:

a) El ordenador mostrará una tirada que consiste en mostrar 3 figuras. Hay 5 figuras posibles: **corazón**, **diamante**, **herradura**, **campana** y **limón**.

b) Si las tres figuras son diferentes se debe mostrar el mensaje **“Lo siento, ha perdido”**.

c) Si hay dos figuras iguales y una diferente se debe mostrar el mensaje **“Bien, ha recuperado su moneda”**.

d) Si las tres figuras son iguales se debe mostrar **“Enhorabuena, ha ganado 10 monedas”**.

Ejemplo 1:

```
diamante diamante limón
Bien, ha recuperado su moneda
```

Ejemplo 2:

```
herradura campana diamante
Lo siento, ha perdido
```

Ejemplo 3:

```
corazón corazón corazón
Enhorabuena, ha ganado 10 monedas
```



Ejercicio 17

Realiza un programa que pinte por pantalla una pecera con un pececito dentro. Se debe pedir al usuario el ancho y el alto de la pecera, que como mínimo serán de 4 unidades. No hay que comprobar que los datos se introducen correctamente; podemos suponer que el usuario los introduce bien. Dentro de la pecera hay que colocar de forma aleatoria un pececito, que puede estar situado en cualquiera de las posiciones que quedan en el hueco que forma el rectángulo.

Ejemplo:

```
Por favor, introduzca la altura de la pecera (como mínimo 4): 4
Ahora introduzca la anchura (como mínimo 4): 7

* * * * * * *
*           *
*       &   *
* * * * * * *
```



Ejercicio 18

Sinestesio y Casilda van a pintar los tres dormitorios de su casa, quieren sustituir el color blanco por colores más alegres. Realiza un programa que genere de forma aleatoria una secuencia de tres colores aleatorios (uno para cada dormitorio) de tal forma que no se repita ninguno. Los colores entre los que debe elegir el programa son los siguientes: rojo, azul, verde, amarillo, violeta y naranja.



Ejercicio 19

Escribe un programa que muestre 50 números enteros aleatorios comprendidos entre el -100 y el 200 ambos incluidos y separados por espacios. Muestra luego el máximo de los pares el mínimo de los impares y la media de todos los números generados.



Ejercicio 20

Realiza un programa que pinte por pantalla una cuba con cierta cantidad de agua. La capacidad será indicada por el usuario. La cuba se llenará con una cantidad aleatoria de agua que puede ir entre 0 y la capacidad máxima que pueda admitir. El ancho de la cuba no varía.

Ejemplo:

```
Por favor, indique la capacidad de la cuba en litros: 3

*    *
*====*
*====*
******
La cuba tiene una capacidad de 3 litros y contiene 2 litros de agua.
```



Ejercicio 21

Realiza un programa que genere una secuencia de cinco monedas de curso legal lanzadas al aire. Las monedas disponibles son de 1 céntimo, 2 céntimos, 5 céntimos, 10 céntimos, 20 céntimos, 50 céntimos, 1 euro y 2 euros. Las dos posiciones posibles son cara y cruz.

Ejemplo:

```
2 céntimos - cara
20 céntimos - cruz
50 céntimos - cruz
1 euro - cruz
2 euros - cara
```



Ejercicio 22

Realiza un programa que pinte por pantalla una serpiente con un "serpenteo" aleatorio. La cabeza se representará con el carácter @ y se debe colocar exactamente en la posición 13 (con 12 espacios delante). A partir de ahí, el cuerpo irá serpenteando de la siguiente manera: se generará de forma aleatoria un valor entre tres posibles que hará que el siguiente carácter se coloque una posición a la izquierda del anterior, alineado con el anterior o una posición a la derecha del anterior. La longitud de la serpiente se pedirá por teclado y se supone que el usuario introducirá un dato correcto.

Ejemplo:

```
Por favor, introduzca la longitud de la serpiente en caracteres contando la cabeza: 6

      @
      *
     *
      *
      *
       *
```



Ejercicio 23

Las caras de un dado de poker tienen las siguientes figuras: As, K, Q, J, 7 y 8. Escribe un programa que genere de forma aleatoria la tirada de cinco dados.

Ejemplo 1:

```
Q J 7 J As
```

Ejemplo 2:

```
K 8 J As 7
```



Ejercicio 24

Escribe un programa que, dado un número introducido por teclado, elija al azar uno de sus dígitos.

Ejemplo 1:

```
Por favor, introduzca un número entero positivo: 406783
7
```

Ejemplo 2:

```
Por favor, introduzca un número entero positivo: 406783
3
```

Ejemplo 3:

```
Por favor, introduzca un número entero positivo: 406783
0
```



Ejercicio 25

Escribe un programa que muestre por pantalla 100 números enteros separados por un espacio. Los números deben estar generados de forma aleatoria en un rango entre 10 y 200 ambos incluidos. Los primos deben aparecer entre almohadillas (p. ej. #19#) y los múltiplos de 5 entre corchetes (p. ej. [25]).



Ejercicio 26

Realiza un programa que pinte una tableta de turrón con un bocado realizado de forma aleatoria. El ancho y el alto de la tableta se pide por teclado. El bocado se da alrededor del turrón, obviamente no se puede dar un bocado por en medio de la tableta.

Ejemplo 1:

```
Introduzca la anchura de la tableta: 6
Introduzca la altura de la tableta: 4

******
*****
******
******
```

Ejemplo 2:

```
Introduzca la anchura de la tableta: 7
Introduzca la altura de la tableta: 3

*******
*******
 ******
```

Ejemplo 3:

```
Introduzca la anchura de la tableta: 5
Introduzca la altura de la tableta: 5

** **
*****
*****
*****
*****
```



Ejercicio 27

Implementa el juego piedra, papel y tijera. Primero, el usuario introduce su jugada y luego el ordenador genera al azar una de las opciones. Si el usuario introduce una opción incorrecta, el programa deberá mostrar un mensaje de error.

Ejemplo 1:

```
Turno del jugador (introduzca piedra, papel o tijera): papel
Turno del ordenador: papel
Empate
```

Ejemplo 2:

```
Turno del jugador (introduzca piedra, papel o tijera): papel
Turno del ordenador: tijera
Gana el ordenador
```

Ejemplo 3:

```
Turno del jugador (introduzca piedra, papel o tijera): piedra
Turno del ordenador: tijera
Gana el jugador
```



Ejercicio 28

Realiza un programa que sea capaz de recolocar los números de un array de fuera hacia adentro. En primer lugar, el programa pedirá al usuario el tamaño del array. A continuación generará un array con ese tamaño con números enteros aleatorios entre 0 y 200 ambos incluidos. Seguidamente el programa irá colocando desde fuera hacia adentro los números de tal forma que el primero se coloca en la primera posición, el segundo en la última, el tercero en la segunda, el cuarto en la penúltima, el quinto en la tercera, en sexto en la antepenúltima, etc. Se debe mostrar por pantalla tanto el array original como el array resultado.

Ejemplo 1:

```
Introduzca el tamaño del array: 12

Array original:
Índice     0    1    2    3    4    5    6    7    8    9   10   11
Valor    120  148   40  108  188   94   60   65  152   27  121   79

Array resultado:
Índice     0    1    2    3    4    5    6    7    8    9   10   11
Valor    120   40  188   60  152  121   79   27   65   94  108  148
```

Ejemplo 2:

```
Introduzca el tamaño del array: 7

Array original:
Índice     0    1    2    3    4    5    6
Valor    130   36   93  188   20  126   36

Array resultado:
Índice     0    1    2    3    4    5    6
Valor    130   93   20   36  126  188   36
```



Ejercicio 29

Realiza un programa que muestre la previsión del tiempo para mañana en Málaga. Las temperaturas máxima y mínima se deben generar de forma aleatoria entre los intervalos máximos y mínimos absolutos medidos en las últimas décadas para cada estación. La probabilidad de que esté soleado o nublado en cada estación se proporciona a continuación. Obviamente, la temperatura mínima deberá ser menor o igual que la temperatura máxima.

	Primavera	Verano	Otoño	Invierno
Temperatura mínima absoluta	15	25	20	0
Temperatura máxima absoluta	30	45	30	25
Probabilidad de soleado-nublado	60% - 40%	80% - 20%	40% - 60%	20% - 80%

Ejemplo:

```
1. Primavera
2. Verano
3. Otoño
4. Invierno
Seleccione la estación del año (1-4): 4

Previsión del tiempo para mañana
--------------------------------
Temperatura mínima: 10ºC
Temperatura máxima: 16ºC
Nublado
```



Ejercicio 30

El pequeño Roberto tenía como mascota un pececillo dentro de una pecera. Los Reyes Magos le han traído un caballito de mar ($) y una caracola (@) para que le hagan compañía al pez. Realiza un programa que pinte por pantalla la pecera con los tres animalitos acuáticos colocados dentro en posiciones aleatorias. Por una cuestión de física elemental, **ninguno de los animales puede coincidir en la misma posición**. Se debe pedir al usuario el ancho y el alto de la pecera, que como mínimo serán de 4 unidades.

Ejemplo:

```
Por favor, introduzca la altura de la pecera (como mínimo 4): 4
Ahora introduzca la anchura (como mínimo 4): 7

* * * * * * *
*   @     & *
*     $     *
* * * * * * *
```



Ejercicio 31

Realiza el juego del **“Craps”**. Las reglas son las siguientes: Al comenzar la partida, el jugador introduce la cantidad de dinero que quiere apostar. Se muestra la tirada aleatoria de dos dados. Si entre los dos dados suman 7 u 11, el jugador gana la misma cantidad que apostó y termina la partida. Por ej. si apostó 1000 €, gana otros 1000 € y acaba con 2000 €. Si entre los dos dados suman 2, 3 o 12, el jugador pierde todo su dinero y termina la partida. Si no se da ninguno de los casos anteriores, es decir si sale 4, 5, 6, 8, 9 o 10, el juego entra en una segunda etapa. En esta etapa, el jugador buscará volver a obtener ese número en los dados. Si consigue repetir ese número, gana. Si sale un 7, pierde. Si sale otro número, tiene que seguir tirando.



Ejercicio 32

Realiza un programa que pinte un sendero aleatorio. Los bordes se pintan con el carácter “|”. La anchura del sendero siempre es la misma, los dos caracteres del borde más cuatro caracteres en medio, en total 6 caracteres (incluyendo espacios). A cada metro, el sendero puede continuar recto, girar un carácter a la izquierda o girar un carácter a la derecha, por supuesto de forma aleatoria. Por cada metro de sendero - representado por una línea - puede que haya un obstáculo o puede que no, la probabilidad es del 50%. La posición del obstáculo es aleatoria dentro de la línea. En caso de existir un obstáculo en un metro de sendero (en una línea), puede ser una planta (carácter *) o una piedra (carácter O), la probabilidad de que salga uno u otro es la misma. Recuerda que nunca habrá más de un obstáculo por metro de sendero, habrá uno o ninguno.

Ejemplo 1:

```
Introduzca la longitud del sendero en metros: 7

      | *  |
       |   O|
       |    |
      |  * |
       |*   |
       |  O |
        |    |
```

Ejemplo 2:

```
Introduzca la longitud del sendero en metros: 5

      |    |
      |   O|
     |    |
```

```
|   *  |
  |*     |
```

7. Arrays

7.1 Arrays de una dimensión

Un *array* es un tipo de dato capaz de almacenar múltiples valores. Se utiliza para agrupar datos muy parecidos, por ejemplo, si se necesita almacenar la temperatura media diaria en Málaga durante el último año se pueden utilizar las variables `temp0`, `temp1`, `temp2`, `temp3`, `temp4`, … y así hasta 365 variables distintas pero sería poco práctico; es mejor utilizar un *array* de nombre `temp` y usar un índice para referenciar la temperatura de un día concreto del año.

En matemáticas, un *array* de una dimensión se llama vector. En este libro no utilizaremos este término para no confundirlo con la clase `Vector` de Java.

Veamos con un ejemplo cómo se crea y se utiliza un *array*.

```java
/**
 * Ejemplo de uso de arrays
 *
 * @author Luis José Sánchez
 */
public class Array01 {
  public static void main(String[] args) {

    int[] n; // se define n como un array de enteros
    n = new int[4]; // se reserva espacio para 4 enteros

    n[0] = 26;
    n[1] = -30;
    n[2] = 0;
    n[3] = 100;

    System.out.print("Los valores del array n son los siguientes: ");
    System.out.print(n[0] + ", " + n[1] + ", " + n[2] + ", " + n[3]);

    int suma = n[0] + n[3];
    System.out.println("\nEl primer elemento del array más el último suman " + suma);
  }
}
```

Observa que el índice de cada elemento de un *array* se indica entre corchetes de tal forma que `n[3]` es el cuarto elemento ya que el primer índice es el 0.

Array n

Índice	Valor
0	26
1	-30
2	0
3	11

Fíjate que la definición del *array* y la reserva de memoria para los cuatro elementos que la componen se ha realizado en dos líneas diferentes.

```
int[] n; // se define n como un array de números enteros
n = new int[4]; // se reserva espacio en memoria para 4 números enteros
```

Normalmente se abrevian estas dos líneas para quedarse en una sola. A efectos prácticos es exactamente lo mismo.

```
int[] x = new int[5];
```

El programa completo quedaría como se muestra a continuación.

```
/**
 * Ejemplo de uso de arrays
 *
 * @author Luis José Sánchez
 */
public class Array02 {
  public static void main(String[] args) {

    // definición del array y reserva de memoria en la misma línea
    int[] x = new int[5];

    x[0] = 8;
    x[1] = 33;
    x[2] = 200;
    x[3] = 150;
    x[4] = 11;

    System.out.println("El array x tiene 5 elementos ¿cuál de ellos quiere ver?");
    System.out.print("Introduzca un número del 0 al 4: ");
    int indice = Integer.parseInt(System.console().readLine());
    System.out.print("El elemento que se encuentra en la posición " + indice);
    System.out.println(" es el " + x[indice]);
  }
}
```

El *array* `x` del ejemplo anterior se ha ido rellenando elemento a elemento. Si se conocen previamente todos los valores iniciales del *array*, se puede crear e inicializar en una sola línea.

```
int[] x = {8, 33, 200, 150, 11};
```

Cada elemento del *array* se puede utilizar exactamente igual que cualquier otra variable, es decir, se le puede asignar un valor o se puede usar dentro de una expresión. En el siguiente ejemplo se muestran varias operaciones en las que los operandos son elementos del *array* `num`.

Para recorrer todos los elementos de un *array* se suele utilizar un bucle `for` junto con un índice que va desde 0 hasta el tamaño del *array* menos 1.

```
/**
 * Ejemplo de uso de arrays
 *
 * @author Luis José Sánchez
 */
public class Array03 {
  public static void main(String[] args) {

    int[] num = new int[10];

    num[0] = 8;
    num[1] = 33;
    num[2] = 200;
    num[3] = 150;
    num[4] = 11;
    num[5] = 88;
    num[6] = num[2] * 10;
    num[7] = num[2] / 10;
    num[8] = num[0] + num[1] + num[2];
    num[9] = num[8];

    System.out.println("El array num contiene los siguientes elementos:");

    for (int i = 0; i < 10; i++) {
      System.out.println(num[i]);
    }
  }
}
```

En Java, a diferencia de otros lenguajes como Ruby o PHP, todos los elementos de un *array* deben ser del mismo tipo; por ejemplo, no puede haber un entero en la posición 2 y una cadena de caracteres en la posición 7 del mismo *array*. En el siguiente ejemplo se muestra un *array* de caracteres.

```
/**
 * Ejemplo de uso de arrays
 *
 * @author Luis José Sánchez
 */
public class Array04 {
  public static void main(String[] args) {

    char[] caracter = new char[6];

    caracter[0] = 'R';
    caracter[1] = '%';
    caracter[2] = '&';
    caracter[3] = '+';
    caracter[4] = 'A';
    caracter[5] = '2';

    System.out.println("El array caracter contiene los siguientes elementos:");

    for (int i = 0; i < 6; i++) {
      System.out.println(caracter[i]);
    }
  }
}
```

En el siguiente ejemplo se muestra un *array* de números de tipo `double` que almacena notas de alumnos.

```
/**
 * Ejemplo de uso de arrays
 *
 * @author Luis José Sánchez
 */
public class Array05 {

  public static void main(String[] args) {

    double[] nota = new double[4];

    System.out.println("Para calcular la nota media necesito saber la ");
    System.out.println("nota de cada uno de tus exámenes.");

    for (int i = 0; i < 4; i++) {
      System.out.print("Nota del examen nº " + (i + 1) + ": ");
      nota[i] = Double.parseDouble(System.console().readLine());
    }
```

```
    System.out.println("Tus notas son: ");

    double suma = 0;

    for (int i = 0; i < 4; i++) {
      System.out.print(nota[i] + "  ");
      suma += nota[i];
    }

    System.out.println("\nLa media es " + suma / 4);
  }
}
```

Veamos otro ejemplo. Esta vez usamos un *array* de cadenas de caracteres para almacenar colores.

```
String[] color = {"rojo", "amarillo", "verde", "blanco", "azul", "negro"};
```

El valor de `color[0]` sería `rojo`, el de `color[1]` sería `amarillo`, etc.

El programa genera una bandera de colores aleatorios. Observa que se utiliza como índice `(int)(Math.random() * 6)`, que es un número aleatorio entre 0 y 5.

```
import java.util.Scanner;

/**
 * Ejemplo de uso de arrays
 *
 * @author Luis José Sánchez
 */
public class Array06GeneradorAleatorioDeBanderas {
  public static void main(String[] args) {
    String[] color = {"rojo", "amarillo", "verde", "blanco", "azul", "negro"};

    System.out.println("Generador aleatorio de banderas");

    Scanner s = new Scanner(System.in);

    System.out.print("¿Cuántas franjas quiere para la bandera? ");
    int franjas = Integer.parseInt(s.nextLine());

    System.out.println("--------------");
    for (int i = 0; i < franjas; i++) {
      System.out.println(color[(int)(Math.random() * 6)]);
      System.out.println("--------------");
```

```
    }
  }
}
```

7.2 Arrays bidimensionales

Un *array* bidimensional utiliza dos índices para localizar cada elemento. Podemos ver este tipo de dato como un *array* que, a su vez, contiene otros *arrays*. También se puede ver como una cuadrícula en la que los datos quedan distribuidos en filas y columnas.

En el siguiente ejemplo se muestra la definición y el uso de un *array* de dos dimensiones.

```
/**
 * @author Luis José Sánchez
 *
 * Ejemplo de uso de arrays bidimensionales
 */
public class ArrayBi01 {
  public static void main(String[] args)
    throws InterruptedException { // Se añade esta línea para poder usar sleep

      int[][] n = new int[3][2]; // array de 3 filas por 2 columnas

      n[0][0]=20;
      n[1][0]=67;
      n[1][1]=33;
      n[2][1]=7;

      int fila, columna;

      for(fila = 0; fila < 3; fila++) {

        System.out.print("Fila: " + fila);

        for(columna = 0; columna < 2; columna++) {
          System.out.printf("%10d ", n[fila][columna]);
          Thread.sleep(1000); // retardo de un segundo
        }
        System.out.println();
      }
  }
}
```

Mediante la línea `int[][] n = new int[3][2]` se define un *array* bidimensional de 3 filas por 2 columnas, pero bien podrían ser 2 filas por 3 columnas, según el objetivo y el uso que del *array* haga el programador.

Los valores del *array* bidimensional se pueden proporcionar en la misma línea de la definición como se muestra en el siguiente ejemplo.

```
/**
 * @author Luis José Sánchez
 *
 * Ejemplo de uso de arrays bidimensionales
 */
public class ArrayBi02 {
  public static void main(String[] args)
    throws InterruptedException { // Se añade esta línea para poder usar sleep

      int fila, columna;
      int[][] n = {{20, 4}, {67, 33}, {0,7}};

      for(fila = 0; fila < 3; fila++) {

        System.out.print("Fila: " + fila);

        for(columna = 0; columna < 2; columna++) {
          System.out.printf("%10d ", n[fila][columna]);
          Thread.sleep(1000); // retardo de un segundo
        }
        System.out.println();
      }
  }
}
```

Los *arrays* bidimensionales se utilizan con frecuencia para situar objetos en un plano como por ejemplo las piezas de ajedrez en un tablero, o un personaje de video-juego en un laberinto.

En el siguiente programa se colocan una mina y un tesoro de forma aleatoria en un cuadrante de cuatro filas por cinco columnas. El usuario intentará averiguar dónde está el tesoro indicando las coordenadas (x, y).

```java
/**
 * Minijuego "Busca el tesoro"
 *
 * Se colocan una mina y un tesoro de forma aleatoria en un cuadrante de
 * cuatro filas por cinco columnas. El usuario intentará averiguar dónde
 * está el tesoro.
 *
 * @author Luis José Sánchez
 */
public class BuscaTesoro {
  public static void main(String[] args) {

    // se definen constantes para representar el
    // contenido de las celdas
    final int VACIO = 0;
    final int MINA = 1;
    final int TESORO = 2;
    final int INTENTO = 3;


    int x;
    int y;
    int[][] cuadrante = new int[5][4];

    // inicializa el array
    for(x = 0; x < 4; x++) {
      for(y = 0; y < 3; y++) {
        cuadrante[x][y] = VACIO;
      }
    }
    // coloca la mina
    int minaX = (int)(Math.random() * 5);
    int minaY = (int)(Math.random() * 4);
    cuadrante[minaX][minaY] = MINA;

    // coloca el tesoro
    int tesoroX;
    int tesoroY;
    do {
      tesoroX = (int)(Math.random() * 5);
      tesoroY = (int)(Math.random() * 4);
    } while ((minaX == tesoroX) && (minaY == tesoroY));
    cuadrante[tesoroX][tesoroY] = TESORO;

    // juego
    System.out.println("¡BUSCA EL TESORO!");
```

```
boolean salir = false;
String c = "";
do {
  // pinta el cuadrante
  for(y = 3; y >= 0; y--) {
    System.out.print(y + "|");
    for(x = 0; x < 5; x++) {
      if (cuadrante[x][y] == INTENTO) {
        System.out.print("X ");
      } else {
        System.out.print("  ");
      }
    }
    System.out.println();
  }
  System.out.println("  ----------\n  0 1 2 3 4\n");

  // pide las coordenadas
  System.out.print("Coordenada x: ");
  x = Integer.parseInt(System.console().readLine());
  System.out.print("Coordenada y: ");
  y = Integer.parseInt(System.console().readLine());

  // mira lo que hay en las coordenadas indicadas por el usuario
  switch(cuadrante[x][y]) {
    case VACIO:
      cuadrante[x][y] = INTENTO;
      break;
    case MINA:
      System.out.println("Lo siento, has perdido.");
      salir = true;
      break;
    case TESORO:
      System.out.println("¡Enhorabuena! ¡Has encontrado el tesoro!");
      salir = true;
      break;
    default:
  }
} while (!salir);

// pinta el cuadrante
for(y = 3; y >= 0; y--) {
  System.out.print(y + " ");
  for(x = 0; x < 5; x++) {
    switch(cuadrante[x][y]) {
      case VACIO:
```

```
            c = "  ";
            break;
          case MINA:
            c = "* ";
            break;
          case TESORO:
            c = "€ ";
            break;
          case INTENTO:
            c = "X ";
            break;
          default:
        }
        System.out.print(c);
      }
      System.out.println();
    }
    System.out.println("  ----------\n  0 1 2 3 4\n");
  }
}
```

7.3 Recorrer arrays con `for` al estilo `foreach`

Al trabajar con *arrays* es muy frecuente cometer errores utilizando los índices. El error más típico consiste en intentar acceder a un elemento mediante un índice que se sale de los límites. Por ejemplo, si tenemos el *array* `n` definido de la siguiente forma `int[] n = new int[10]`, cuando intentamos acceder a `n[-1]` o a `n[10]` obtenemos un error en tiempo de ejecución.

Para recorrer un *array* de un modo más práctico y sencillo, sin que tengamos que preocuparnos de los límites, podemos utilizar el bucle `for` con el formato `foreach`. De esta forma indicamos simplemente el nombre del *array* que queremos recorrer y en qué variable se va a ir colocando cada elemento con cada iteración del bucle. No hay que especificar con qué índice comienza y termina el bucle, de eso se encarga Java.

A continuación se muestra el ejemplo `Array05.java` visto anteriormente pero, esta vez, utilizando el `for` a la manera `foreach`.

```
/**
 * Recorre un array con un for al estilo foreach.
 *
 * @author Luis José Sánchez
 */
public class ArrayForEach {
    public static void main(String[] args) {

    double[] nota = new double[4];

    System.out.println("Para calcular la nota media necesito saber la ");
    System.out.println("nota de cada uno de tus exámenes.");

    for (int i = 0; i < 4; i++) {
      System.out.print("Nota del examen nº " + (i + 1) + ": ");
      nota[i] = Double.parseDouble(System.console().readLine());
    }

    System.out.println("Tus notas son: ");

    double suma = 0;

    for (double n : nota) { // for al estilo foreach
      System.out.print(n + "  ");
      suma += n;
    }
    System.out.println("\nLa media es " + suma / 4);
  }
}
```

Fíjate en el segundo `for`; en este caso no se utiliza ningún índice; simplemente decimos "ve sacando uno a uno los elementos del array `nota` y deposita cada uno de esos elementos en la variable `n` que es de tipo `double`".

7.4 Ejercicios

Arrays de una dimensión



Ejercicio 1

Define un *array* de 12 números enteros con nombre `num` y asigna los valores según la tabla que se muestra a continuación. Muestra el contenido de todos los elementos del *array*. ¿Qué sucede con los valores de los elementos que no han sido inicializados?

Índice	0	1	2	3	4	5	6	7	8	9	10	11
Valor	39	-2			0		14		5	120		



Ejercicio 2

Define un *array* de 10 caracteres con nombre `simbolo` y asigna valores a los elementos según la tabla que se muestra a continuación. Muestra el contenido de todos los elementos del *array*. ¿Qué sucede con los valores de los elementos que no han sido inicializados?

Índice	0	1	2	3	4	5	6	7	8	9
Valor	'a'	'x'			'@'		' '	'+'	'Q'	



Ejercicio 3

Escribe un programa que lea 10 números por teclado y que luego los muestre en orden inverso, es decir, el primero que se introduce es el último en mostrarse y viceversa.



Ejercicio 4

Define tres *arrays* de 20 números enteros cada una, con nombres `numero`, `cuadrado` y `cubo`. Carga el *array* `numero` con valores aleatorios entre 0 y 100. En el *array* `cuadrado` se deben almacenar los cuadrados de los valores que hay en el *array* `numero`. En el *array* `cubo` se deben almacenar los cubos de los valores que hay en `numero`. A continuación, muestra el contenido de los tres *arrays* dispuesto en tres columnas.



Ejercicio 5

Escribe un programa que pida 10 números por teclado y que luego muestre los números introducidos junto con las palabras “máximo” y “mínimo” al lado del máximo y del mínimo respectivamente.



Ejercicio 6

Escribe un programa que lea 15 números por teclado y que los almacene en un *array*. Rota los elementos de ese *array*, es decir, el elemento de la posición 0 debe pasar a la posición 1, el de la 1 a la 2, etc. El número que se encuentra en la última posición debe pasar a la posición 0. Finalmente, muestra el contenido del *array*.



Ejercicio 7

Escribe un programa que genere 100 números aleatorios del 0 al 20 y que los muestre por pantalla separados por espacios. El programa pedirá entonces por teclado dos valores y a continuación cambiará todas las ocurrencias del primer valor por el segundo en la lista generada anteriormente. Los números que se han cambiado deben aparecer entrecomillados.



Ejercicio 8

Realiza un programa que pida la temperatura media que ha hecho en cada mes de un determinado año y que muestre a continuación un diagrama de barras horizontales con esos datos. Las barras del diagrama se pueden dibujar a base de asteriscos o cualquier otro carácter.



Ejercicio 9

Realiza un programa que pida 8 números enteros y que luego muestre esos números junto con la palabra “par” o “impar” según proceda.



Ejercicio 10

Escribe un programa que genere 20 números enteros aleatorios entre 0 y 100 y que los almacene en un *array*. El programa debe ser capaz de pasar todos los números pares a las primeras posiciones del *array* (del 0 en adelante) y todos los números impares a las celdas restantes. Utiliza *arrays* auxiliares si es necesario.



Ejercicio 11

Realiza un programa que pida 10 números por teclado y que los almacene en un *array*. A continuación se mostrará el contenido de ese *array* junto al índice (0 – 9) utilizando para ello una tabla. Seguidamente el programa pasará los primos a las primeras posiciones, desplazando el resto de números (los que no son primos) de tal forma que no se pierda ninguno. Al final se debe mostrar el *array* resultante.

Por ejemplo:

Índice	0	1	2	3	4	5	6	7	8	9
Valor	20	5	7	4	32	9	2	14	11	6

Array inicial

Índice	0	1	2	3	4	5	6	7	8	9
Valor	5	7	2	11	20	4	32	9	14	6

Array final



Ejercicio 12

Realiza un programa que pida 10 números por teclado y que los almacene en un *array*. A continuación se mostrará el contenido de ese *array* junto al índice (0 – 9). Seguidamente el programa pedirá dos posiciones a las que llamaremos "inicial" y "final". Se debe comprobar que inicial es menor que final y que ambos números están entre 0 y 9. El programa deberá colocar el número de la posición inicial en la posición final, rotando el resto de números para que no se pierda ninguno. Al final se debe mostrar el *array* resultante.

Por ejemplo, para inicial = 3 y final = 7:

Índice	0	1	2	3	4	5	6	7	8	9
Valor	20	5	7	4	32	9	2	14	11	6

Array inicial

Índice	0	1	2	3	4	5	6	7	8	9
Valor	6	20	5	7	32	9	2	4	14	11

Array final



Ejercicio 13

Escribe un programa que rellene un array de 100 elementos con números enteros aleatorios comprendidos entre 0 y 500 (ambos incluidos). A continuación el programa mostrará el array y preguntará si el usuario quiere destacar el máximo o el mínimo. Seguidamente se volverá a mostrar el array escribiendo el número destacado entre dobles asteriscos.

Ejemplo:

```
459 204 20 250 178 90 353 32 229 357 224 454 260 310 140 249 332 426 423 413 96
447 465 298 459 411 118 480 302 417 42 82 126 82 474 362 76 190 104 21 257 88 21
251 6 383 47 78 392 394 244 494 87 253 376 379 98 364 237 13 299 228 409 402 225
426 267 330 243 209 426 435 309 356 173 130 416 15 477 34 28 377 193 481 368 466
262 422 275 384 399 397 87 218 84 312 480 207 68 108

¿Qué quiere destacar? (1 – mínimo, 2 – máximo): 1

459 204 20 250 178 90 353 32 229 357 224 454 260 310 140 249 332 426 423 413 96
447 465 298 459 411 118 480 302 417 42 82 126 82 474 362 76 190 104 21 257 88 21
251 **6** 383 47 78 392 394 244 494 87 253 376 379 98 364 237 13 299 228 409 402
225 426 267 330 243 209 426 435 309 356 173 130 416 15 477 34 28 377 193 481 368
466 262 422 275 384 399 397 87 218 84 312 480 207 68 108
```



Ejercicio 14

Escribe un programa que pida 8 palabras y las almacene en un array. A continuación, las palabras correspondientes a colores se deben almacenar al comienzo y las que no son colores a continuación. Puedes utilizar tantos arrays auxiliares como quieras. Los colores que conoce el programa deben estar en otro array y son los siguientes: verde, rojo, azul, amarillo, naranja, rosa, negro, blanco y morado.

Ejemplo:

Array original:

0	1	2	3	4	5	6	7
casa	azul	verde	orden	morado	bombilla	bici	rosa

Array resultado:

0	1	2	3	4	5	6	7

```
┌────────┬────────┬────────┬────────┬────────┬────────┬────────┬────────┐
│azul    │verde   │morado  │rosa    │casa    │orden   │bombilla│bici    │
└────────┴────────┴────────┴────────┴────────┴────────┴────────┴────────┘
```



Ejercicio 15

Un restaurante nos ha encargado una aplicación para colocar a los clientes en sus mesas. En una mesa se pueden sentar de 0 (mesa vacía) a 4 comensales (mesa llena). Cuando llega un cliente se le pregunta cuántos son. De momento el programa no está preparado para colocar a grupos mayores a 4, por tanto, si un cliente dice por ejemplo que son un grupo de 6, el programa dará el mensaje **"Lo siento, no admitimos grupos de 6, haga grupos de 4 personas como máximo e intente de nuevo"**. Para el grupo que llega, se busca siempre la primera mesa libre (con 0 personas). Si no quedan mesas libres, se busca donde haya un hueco para todo el grupo, por ejemplo si el grupo es de dos personas, se podrá colocar donde haya una o dos personas. Inicialmente, las mesas se cargan con valores aleatorios entre 0 y 4. Cada vez que se sientan nuevos clientes se debe mostrar el estado de las mesas. Los grupos no se pueden romper aunque haya huecos sueltos suficientes. El funcionamiento del programa se ilustra a continuación:

Ejemplo:

```
┌─────────┬────┬────┬────┬────┬────┬────┬────┬────┬────┬────┐
│Mesa nº  │  1 │  2 │  3 │  4 │  5 │  6 │  7 │  8 │  9 │ 10 │
├─────────┼────┼────┼────┼────┼────┼────┼────┼────┼────┼────┤
│Ocupación│  3 │  2 │  0 │  2 │  4 │  1 │  0 │  2 │  1 │  1 │
└─────────┴────┴────┴────┴────┴────┴────┴────┴────┴────┴────┘

¿Cuántos son? (Introduzca -1 para salir del programa): 2
Por favor, siéntense en la mesa número 3.

┌─────────┬────┬────┬────┬────┬────┬────┬────┬────┬────┬────┐
│Mesa nº  │  1 │  2 │  3 │  4 │  5 │  6 │  7 │  8 │  9 │ 10 │
├─────────┼────┼────┼────┼────┼────┼────┼────┼────┼────┼────┤
│Ocupación│  3 │  2 │  2 │  2 │  4 │  1 │  0 │  2 │  1 │  1 │
└─────────┴────┴────┴────┴────┴────┴────┴────┴────┴────┴────┘

¿Cuántos son? (Introduzca -1 para salir del programa): 4
Por favor, siéntense en la mesa número 7.

┌─────────┬────┬────┬────┬────┬────┬────┬────┬────┬────┬────┐
│Mesa nº  │  1 │  2 │  3 │  4 │  5 │  6 │  7 │  8 │  9 │ 10 │
├─────────┼────┼────┼────┼────┼────┼────┼────┼────┼────┼────┤
│Ocupación│  3 │  2 │  2 │  2 │  4 │  1 │  4 │  2 │  1 │  1 │
```

```
└──────────┴─────┴─────┴─────┴─────┴─────┴─────┴─────┴─────┴─────┴─────┘

¿Cuántos son? (Introduzca -1 para salir del programa): 3
Tendrán que compartir mesa. Por favor, siéntense en la mesa número 6.

┌──────────┬─────┬─────┬─────┬─────┬─────┬─────┬─────┬─────┬─────┬─────┐
│Mesa nº   │  1  │  2  │  3  │  4  │  5  │  6  │  7  │  8  │  9  │ 10  │
├──────────┼─────┼─────┼─────┼─────┼─────┼─────┼─────┼─────┼─────┼─────┤
│Ocupación│  3  │  2  │  2  │  2  │  4  │  4  │  4  │  2  │  1  │  1  │
└──────────┴─────┴─────┴─────┴─────┴─────┴─────┴─────┴─────┴─────┴─────┘

¿Cuántos son? (Introduzca -1 para salir del programa): 4
Lo siento, en estos momentos no queda sitio.

┌──────────┬─────┬─────┬─────┬─────┬─────┬─────┬─────┬─────┬─────┬─────┐
│Mesa nº   │  1  │  2  │  3  │  4  │  5  │  6  │  7  │  8  │  9  │ 10  │
├──────────┼─────┼─────┼─────┼─────┼─────┼─────┼─────┼─────┼─────┼─────┤
│Ocupación│  3  │  2  │  2  │  2  │  4  │  4  │  4  │  2  │  1  │  1  │
└──────────┴─────┴─────┴─────┴─────┴─────┴─────┴─────┴─────┴─────┴─────┘

¿Cuántos son? (Introduzca -1 para salir del programa): -1
Gracias. Hasta pronto.
```



Ejercicio 16

Escribe un programa que rellene un array de 20 elementos con números enteros aleatorios comprendidos entre 0 y 400 (ambos incluidos). A continuación el programa mostrará el array y preguntará si el usuario quiere resaltar los múltiplos de 5 o los múltiplos de 7. Seguidamente se volverá a mostrar el array escribiendo los números que se quieren resaltar entre corchetes.

Ejemplo:

```
159 204 20 250 178 90 353 32 229 357 224 54 260 310 140 249 335 326 223 13
¿Qué números quiere resaltar? (1 - los múltiplos de 5, 2 - los múltiplos de 7): 1
159 204 [20] [250] 178 [90] 353 32 229 357 224 54 [260] [310] [140] 249 [335] 326 223 13
```



Ejercicio 17

Escribe un programa que muestre por pantalla un array de 10 números enteros generados al azar entre 0 y 100. A continuación, el programa debe pedir un número al usuario. Se debe comprobar que el número introducido por teclado se encuentra dentro del array, en caso contrario se mostrará un mensaje por pantalla y se volverá a pedir un número; así hasta que el usuario introduzca uno correctamente. A continuación, el programa rotará el array hacia la derecha las veces que haga falta hasta que el número introducido quede situado en la posición 0 del array. Por último, se mostrará el array rotado por pantalla.



Ejercicio 18

Realiza un programa que genere 10 números enteros aleatorios entre 0 y 200 ambos incluidos y que los almacene en un array. A continuación, el programa debe mostrar el contenido de ese array junto al índice (0 – 9). Seguidamente el programa debe colocar de forma alterna y en orden los menores o iguales de 100 y los mayores de 100: primero menor, luego mayor, luego menor, luego mayor… Cuando se acaben los menores o los mayores, se completará con los números que queden.

Ejemplo 1:

Array original:

Índice	0	1	2	3	4	5	6	7	8	9
Valor	127	178	11	39	121	82	130	47	128	129

Array resultado:

Índice	0	1	2	3	4	5	6	7	8	9
Valor	11	127	39	178	82	121	47	130	128	129

Ejemplo 2:

Array original:

Índice	0	1	2	3	4	5	6	7	8	9
Valor	170	189	87	149	176	110	119	9	33	157

```
Array resultado:
```

Índice	0	1	2	3	4	5	6	7	8	9
Valor	87	170	9	189	33	149	176	110	119	157

Ejemplo 3:

```
Array original:
```

Índice	0	1	2	3	4	5	6	7	8	9
Valor	119	88	139	146	34	4	195	160	27	115

```
Array resultado:
```

Índice	0	1	2	3	4	5	6	7	8	9
Valor	88	119	34	139	4	146	27	195	160	115



Ejercicio 19

Realiza un programa que sea capaz de insertar un número en una posición concreta de un array. En primer lugar, el programa generará un array de 12 números enteros aleatorios entre 0 y 200 ambos incluidos. A continuación se debe mostrar el contenido de ese array junto al índice (0 – 11). Seguidamente el programa preguntará por el número que se quiere insertar y por la posición donde será insertado. Los números del array se desplazan a la derecha para dejar sitio al nuevo. El último número (el que se encuentra en la posición 11) siempre se perderá.

Ejemplo 1:

```
Array original:

Índice    0    1    2    3    4    5    6    7    8    9   10   11
Valor     5   82    0  131  113  200  134   44   48  134   68  151

Introduzca el número que quiere insertar: 77
Introduzca la posición donde lo quiere insertar (0 – 11): 6

Array resultado:
```

```
Índice    0    1    2    3    4    5    6    7    8    9   10   11
Valor     5   82    0  131  113  200   77  134   44   48  134   68
```

Ejemplo 2:

```
Array original:

Índice    0    1    2    3    4    5    6    7    8    9   10   11
Valor   148   86   64  126   77  148  182   99    8  126   73   20

Introduzca el número que quiere insertar: 33
Introduzca la posición donde lo quiere insertar (0 - 11): 11

Array resultado:

Índice    0    1    2    3    4    5    6    7    8    9   10   11
Valor   148   86   64  126   77  148  182   99    8  126   73   33
```

Ejemplo 3:

```
Array original:

Índice    0    1    2    3    4    5    6    7    8    9   10   11
Valor   186    4   58  150  200  141   34  137   62  170  200   70

Introduzca el número que quiere insertar: 88
Introduzca la posición donde lo quiere insertar (0 - 11): 2

Array resultado:

Índice    0    1    2    3    4    5    6    7    8    9   10   11
Valor   186    4   88   58  150  200  141   34  137   62  170  200
```



Ejercicio 20

Implementa un programa que calcule la denominación ordinal de los reyes de una secuencia histórica. El programa solicitará la cantidad de reyes que se van a introducir, y a continuación recibirá los nombres de los reyes. Presentará por pantalla dichos nombres, pero colocándoles el ordinal correspondiente. Así, por ejemplo, si hay dos Felipes en los nombres de los reyes, el primero debería aparecer como Felipe 1º y el segundo como Felipe 2º.

Ejemplo 1:

```
Introduzca el número total de nombres de reyes: 7
Vaya introduciendo los nombres de los reyes y pulsando INTRO.
Felipe
Carlos
Carlos
Fernando
Carlos
Carlos
Felipe

Los reyes introducidos son:
Felipe 1º
Carlos 1º
Carlos 2º
Fernando 1º
Carlos 3º
Carlos 4º
Felipe 2º
```

Ejemplo 2:

```
Introduzca el número total de nombres de reyes: 9
Vaya introduciendo los nombres de los reyes y pulsando INTRO.
Luis
Fernando
Fernando
Carlos
Amadeo
Alfonso
Carlos
Alfonso
Alfonso

Los reyes introducidos son:
Luis 1º
Fernando 1º
Fernando 2º
Carlos 1º
Amadeo 1º
Alfonso 1º
Carlos 2º
Alfonso 2º
Alfonso 3º
```



Ejercicio 21

Escribe un programa que rellene un array de 15 elementos con números enteros comprendidos entre 0 y 500 (ambos incluidos). A continuación, se mostrará el array “cincuerizado”, según el siguiente criterio: si el número que hay en una posición del array es múltiplo de 5, se deja igual, y si no, se cambia por el siguiente múltiplo de 5 que exista a partir de él.

Ejemplo:

```
Array original:
459 204 20 250 178 90 353 35 229 357 224 454 260 310 140

Array cincuerizado:
460 205 20 250 180 90 355 35 230 360 225 455 260 310 140
```

Arrays bidimensionales



Ejercicio 1

Define un *array* de números enteros de 3 filas por 6 columnas con nombre `num` y asigna los valores según la siguiente tabla. Muestra el contenido de todos los elementos del *array* dispuestos en forma de tabla como se muestra en la figura.

Array num	Columna 0	Columna 1	Columna 2	Columna 3	Columna 4	Columna 5
Fila 0	0	30	2			5
Fila 1	75				0	
Fila 2			-2	9		11



Ejercicio 2

Escribe un programa que pida 20 números enteros. Estos números se deben introducir en un *array* de 4 filas por 5 columnas. El programa mostrará las sumas parciales de filas y columnas igual que si de una hoja de cálculo se tratara. La suma total debe aparecer en la esquina inferior derecha.

					Σ fila 0
					Σ fila 0
					Σ fila 0
					Σ fila 0
Σ columna 0	Σ columna 1	Σ columna 2	Σ columna 3	Σ columna 4	**TOTAL**



Ejercicio 3

Modifica el programa anterior de tal forma que los números que se introducen en el *array* se generen de forma aleatoria (valores entre 100 y 999).



Ejercicio 4

Modifica el programa anterior de tal forma que las sumas parciales y la suma total aparezcan en la pantalla con un pequeño retardo, dando la impresión de que el ordenador se queda "pensando" antes de mostrar los números.



Ejercicio 5

Realiza un programa que rellene un *array* de 6 filas por 10 columnas con números enteros positivos comprendidos entre 0 y 1000 (ambos incluidos). A continuación, el programa deberá dar la posición tanto del máximo como del mínimo.



Ejercicio 6

Modifica el programa anterior de tal forma que no se repita ningún número en el *array*.



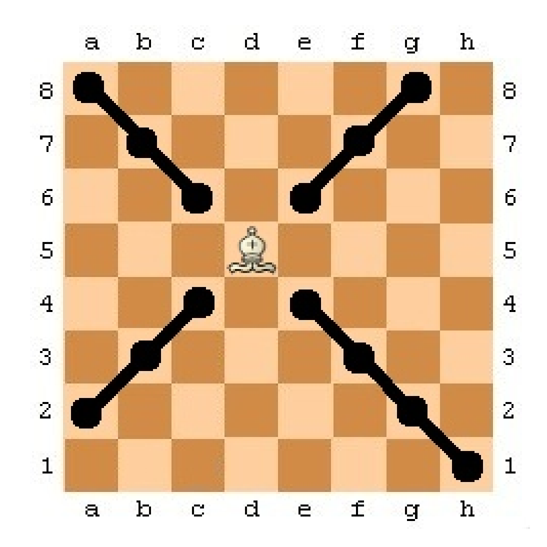
Ejercicio 7

Mejora el juego "Busca el tesoro" de tal forma que si hay una mina a una casilla de distancia, el programa avise diciendo ¡Cuidado! ¡Hay una mina cerca!



Ejercicio 8

Escribe un programa que, dada una posición en un tablero de ajedrez, nos diga a qué casillas podría saltar un alfil que se encuentra en esa posición. Como se indica en la figura, el alfil se mueve siempre en diagonal. El tablero cuenta con 64 casillas. Las columnas se indican con las letras de la "a" a la "h" y las filas se indican del 1 al 8.



Ejemplo:

```
Introduzca la posición del alfil: d5
El álfil puede moverse a las siguientes posiciones:
h1 a2 g2 b3 f3 c4 e4 c6 e6 b7 f7 a8 g8
```



Ejercicio 9

Realiza un programa que sea capaz de rotar todos los elementos de una matriz cuadrada una posición en el sentido de las agujas del reloj. La matriz debe tener 12 filas por 12 columnas y debe contener números generados al azar entre 0 y 100. Se debe mostrar tanto la matriz original como la matriz resultado, ambas con los números convenientemente alineados.



Ejercicio 10

Realiza el juego de las tres en raya.



Ejercicio 11

Realiza un programa que muestre por pantalla un array de 10 filas por 10 columnas relleno con números aleatorios entre 200 y 300. A continuación, el programa debe mostrar los números de la diagonal que va desde la esquina superior izquierda a la esquina inferior derecha, así como el máximo, el mínimo y la media de los números que hay en esa diagonal.



Ejercicio 12

Realiza un programa que muestre por pantalla un array de 9 filas por 9 columnas relleno con números aleatorios entre 500 y 900. A continuación, el programa debe mostrar los números de la diagonal que va desde la esquina inferior izquierda a la esquina superior derecha, así como el máximo, el mínimo y la media de los números que hay en esa diagonal.



Ejercicio 13

Realiza un programa que calcule la estatura media, mínima y máxima en centímetros de personas de diferentes países. El array que contiene los nombres de los paises es el siguiente: **pais = {“España”, “Rusia”, “Japón”, “Australia”}**. Los datos sobre las estaturas se deben simular mediante un array de 4 filas por 10 columnas con números aleatorios generados al azar entre 140 y 210. Los decimales de la media se pueden despreciar. Los nombres de los países se deben mostrar utilizando el array de países (no se pueden escribir directamente).

Ejemplo:

```
                                                  MED MIN MAX
   España: 178 165 148 185 155 141 165 149 155 201 | 164 141 201
    Rusia: 179 189 208 167 186 174 152 192 173 179 | 179 152 179
    Japón: 173 182 168 170 181 197 146 168 166 177 | 172 146 177
Australia: 172 170 187 186 197 143 190 199 187 191 | 182 143 191
```

8. Funciones

8.1 Implementando funciones para reutilizar código

En programación es muy frecuente reutilizar código, es decir, usar código ya existente. Cuando una parte de un programa requiere una funcionalidad que ya está implementada en otro programa no tiene mucho sentido emplear tiempo y energía en implementarla otra vez.

Una función es un trozo de código que realiza una tarea muy concreta y que se puede incluir en cualquier programa cuando hace falta resolver esa tarea. Opcionalmente, las funciones aceptan una entrada (parámetros de entrada) y devuelven una salida.

Observa el siguiente ejemplo. Se trata de un programa que pide un número por teclado y luego dice si el número introducido es o no es primo.

```
/**
 * Dice si un número es o no es primo (sin funciones)
 *
 * @author Luis José Sánchez
 */
public class NumeroPrimo {
  public static void main(String[] args) {

    System.out.print("Introduce un número entero positivo: ");
    int n = Integer.parseInt(System.console().readLine());

    boolean esPrimo = true;
    for (int i = 2; i < n; i++) {
      if ((n % i) == 0) {
        esPrimo = false;
      }
    }

    if (esPrimo) {
      System.out.println("El " + n + " es primo.");
    } else {
      System.out.println("El " + n + " no es primo.");
    }
  }
}
```

Podemos intuir que la tarea de averiguar si un número es o no primo será algo que utilizaremos con frecuencia más adelante así que podemos aislar el trozo de código que realiza ese cometido para usarlo con comodidad en otros programas.

```
/**
 * Dice si un número es o no es primo usando una función
 *
 * @author Luis José Sánchez
 */
public class NumeroPrimoConFuncion {

  // Programa principal //////////////////////////////////////

  public static void main(String[] args) {

    System.out.print("Introduzca un número entero positivo: ");
    int n = Integer.parseInt(System.console().readLine());

    if (esPrimo(n)) {
      System.out.println("El " + n + " es primo.");
    } else {
      System.out.println("El " + n + " no es primo.");
    }
  }

  // Funciones ///////////////////////////////////////////////

  /**
   * Comprueba si un número entero positivo es primo o no.
   * Un número es primo cuando únicamente es divisible entre
   * él mismo y la unidad.
   *
   * @param x un número entero positivo
   * @return  <code>true</code> si el número es primo
   * @return  <code>false</code> en caso contrario
   */
  public static boolean esPrimo(int x) {

    for (int i = 2; i < x; i++) {
      if ((x % i) == 0) {
        return false;
      }
    }

    return true;
  }
```

```
}
```

Cada función tiene una cabecera y un cuerpo. En el ejemplo anterior la cabecera es

```
public static boolean esPrimo(int x)
```

De momento no vamos a explicar qué significa `public static`, lo dejaremos para cuando veamos el capítulo 9 "Programación orientada a objetos", por ahora lo escribiremos tal cual cada vez que definamos una función. A continuación se escribe el tipo de dato que devuelve la función, en este caso es `boolean` porque la función devolverá siempre `true` (verdadero) o `false` falso. Lo último que lleva la cabecera son los parámetros encerrados entre paréntesis. Esos parámetros son los valores que se le pasan a la función para que realice los cálculos. En este caso concreto, el parámetro que se pasa es `x`, o sea, el número que queremos saber si es primo o no. Es necesario indicar siempre el tipo de cada parámetro; en esta ocasión, el parámetro que se pasa es de tipo entero (`int`).

8.2 Comentarios de funciones

Los comentarios tienen un papel muy importante en los programas ya que, aunque no se compilan ni se ejecutan, permiten describir, aclarar o dar información relevante sobre qué hace exactamente el código. Un programa bien comentado es mucho más fácil de corregir, mantener y mejorar que un programa mal comentado o que simplemente no tiene comentarios.

Existe una herramienta llamada `javadoc` que crea unos documentos en HTML con información sobre programas escritos en Java. Para ello, `javadoc` utiliza los comentarios que se han hecho y, además, la meta-información que se incluye en esos comentarios. Esta meta-información es muy fácil de distinguir pues va precedida siempre de un carácter `@`.

Te habrás dado cuenta que en los comentarios de los programas que hemos visto en este manual, se indica el nombre del autor mediante `@author`.

Si los comentarios de los programas en general son importantes, debemos prestar especial atención a los comentarios de las funciones ya que posiblemente serán usadas por otros programadores que querrán saber exactamente cómo se utilizan.

Con la etiqueta `@param` seguida de un nombre de parámetro se indica qué parámetro espera como entrada la función. Si una función acepta varios parámetros, se especificarán varias etiquetas `@param` en los comentarios.

Mediante la etiqueta `@return` especificamos qué devuelve exactamente la función. Aunque el tipo de dato que se devuelve ya viene indicado en la cabecera de la función, mediante la etiqueta `@return` podemos explicarlo con más detalle.

En los comentarios se pueden incluir elementos de HTML como `<code>`, ` `, etc. ya que los entiende perfectamente `javadoc`.

Puedes hacer una prueba generando la documentación del ejemplo ejecutando la siguiente línea en una ventana de terminal. Como tenemos tildes en los comentarios, si queremos que salgan bien en las páginas de documentación, debemos indicar la codificación de caracteres.

```
javadoc -encoding UTF-8 -charset UTF-8 -docencoding UTF-8 NumeroPrimoConFuncion.java
```

Para aprender más sobre `javadoc` puedes consultar la documentación oficial de Oracle[1].

8.3 Creación de bibliotecas de rutinas mediante paquetes

Si la función `esPrimo()` va a ser usada en tres programas diferentes se puede copiar y pegar su código en cada uno de los programas, pero hay una solución mucho más elegante y práctica.

Las funciones de un determinado tipo (por ejemplo funciones matemáticas) se pueden agrupar para crear un paquete (`package`) que luego se importará desde el programa que necesite esas funciones.

Cada paquete se corresponde con un directorio. Por tanto, si hay un paquete con nombre `matematicas` debe haber un directorio llamado también `matematicas` en la misma ubicación del programa que importa ese paquete (normalmente el programa principal).

Las funciones se pueden agrupar dentro de un paquete de dos maneras diferentes. Puede haber subpaquetes dentro de un paquete; por ejemplo, si quisiéramos dividir las funciones matemáticas en funciones relativas al cálculo de áreas y volúmenes de figuras geométricas y funciones relacionadas con cálculos estadísticos, podríamos crear dos directorios dentro de `matematicas` con nombres `geometria` y `estadistica` respectivamente. Estos subpaquetes se llamarían `matematicas.geometria` y `matematicas.estadistica`. Otra manera de agrupar las funciones dentro de un mismo paquete consiste en crear varios ficheros dentro de un mismo directorio. En este caso se podrían crear los ficheros `Geometria.java` y `Estadistica.java`.

Entenderemos mejor todos estos conceptos con un ejemplo completo. Vamos a crear un paquete con nombre `matematicas` que contenga dos clases: `Varias` (para funciones matemáticas de propósito general) y `Geometria`. Por tanto en el disco duro, tendremos una carpeta con nombre `matematicas` que contiene los ficheros `Varias.java` y `Geometria.java`. El contenido de estos ficheros se muestra a continuación.

[1] http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html

```java
package matematicas;

/**
 * Funciones matemáticas de propósito general
 *
 * @author Luis José Sánchez
 */
public class Varias {

  /**
   * Comprueba si un número entero positivo es primo o no.
   * Un número es primo cuando únicamente es divisible entre
   * él mismo y la unidad.
   *
   * @param x un número entero positivo
   * @return  <code>true</code> si el número es primo
   * @return  <code>false</code> en caso contrario
   */
  public static boolean esPrimo(int x) {

    for (int i = 2; i < x; i++) {
      if ((x % i) == 0) {
        return false;
      }
    }

    return true;
  }

  /**
   * Devuelve el número de dígitos que contiene un número entero
   *
   * @param x un número entero
   * @return  la cantidad de dígitos que contiene el número
   */
  public static int digitos(int x) {

    if (x == 0) {
      return 1;
    } else {
      int n = 0;
      while (x > 0) {
        x = x / 10;
        n++;
      }
      return n;
```

```
    }
  }
}
```

Se incluye a continuación `Geometria.java`. Recuerda que tanto `Varias,java` como `Geometria.java` se encuentran dentro del directorio `matematicas`.

```
package matematicas;

/**
 * Funciones geométricas
 *
 * @author Luis José Sánchez
 */
public class Geometria {

  /**
   * Calcula el volumen de un cilindro.
   * Tanto el radio como la altura se deben proporcionar en las
   * mismas unidades para que el resultado sea congruente.
   *
   * @param r radio del cilindro
   * @param h altura del cilindro
   * @return  volumen del cilindro
   */
  public static double volumenCilindro(double r, double h) {
    return Math.PI * r * r * h;
  }

  /**
   * Calcula la longitud de una circunferencia a partir del radio.
   *
   * @param r radio de la circunferencia
   * @return  longitud de la circunferencia
   */
  public static double longitudCircunferencia(double r) {
    return 2 * Math.PI * r;
  }
}
```

Observa que en ambos ficheros se especifica que las clases declaradas (y por tanto las funciones que se definen dentro) pertenecen al paquete `matematicas` mediante la línea `package matematicas`. Ahora ya podemos probar las funciones desde un programa externo. El programa `PruebaFunciones.java` está fuera de la carpeta `matematicas`, justo en un nivel superior en la estructura de directorios.

```java
import matematicas.Varias;
import matematicas.Geometria;

/**
 * Prueba varias funciones
 *
 * @author Luis José Sánchez
 */
public class PruebaFunciones {
  public static void main(String[] args) {

    int n;

    // Prueba esPrimo()

    System.out.print("Introduzca un número entero positivo: ");
    n = Integer.parseInt(System.console().readLine());

    if (matematicas.Varias.esPrimo(n)) {
      System.out.println("El " + n + " es primo.");
    } else {
      System.out.println("El " + n + " no es primo.");
    }


    // Prueba digitos()

    System.out.print("Introduzca un número entero positivo: ");
    n = Integer.parseInt(System.console().readLine());

    System.out.println(n + " tiene " + matematicas.Varias.digitos(n) + " dígitos.");

    // Prueba volumenCilindro()

    double r, h;

    System.out.println("Cálculo del volumen de un cilindro");
    System.out.print("Introduzca el radio en metros: ");
    r = Double.parseDouble(System.console().readLine());
    System.out.print("Introduzca la altura en metros: ");
    h = Double.parseDouble(System.console().readLine());

    System.out.println("El volumen del cilindro es " + matematicas.Geometria.volumenCilindro(r\
, h) + " m3");
  }
}
```

Las líneas

```
import matematicas.Geometria;
import matematicas.Varias;
```

cargan las clases contenidas en el paquete `matematicas` y, por tanto, todas las funciones contenidas en ellas. No vamos a utilizar el comodín de la forma `import matematicas.*;` ya que está desaconsejado su uso en el estándar de codificación en Java de Google. Observa que se escriben por orden alfabético; como el paquete `matematicas` coincide, se escribe antes `Geometria` y luego `Varias`.

El uso de una función es muy sencillo; hay que indicar el nombre del paquete, el nombre de la clase y finalmente el nombre de la función con los parámetros adecuados. Por ejemplo, como vemos en el programa anterior, `matematicas.Varias.digitos(n)` devuelve el número de dígitos de `n`; siendo `matematicas` el paquete, `Varias` la clase, `digitos` la función y `n` el parámetro que se le pasa a la función.

8.4 Ámbito de las variables

El ámbito de una variable es el espacio donde "existe" esa variable o, dicho de otro modo, el contexto dentro del cual la variable es válida.

Seguramente has utilizado muchas veces variables con nombres como `x`, `i`, `max` o `aux`. El ámbito de cada una de esas variables era el programa en el que estaban declaradas. Fíjate que la `x` del primer programa que aparece en este libro no interfiere para nada con la `x` del siguiente programa que usa una variable con ese mismo nombre. Aunque se llamen igual son variables diferentes.

Cuando se implementan funciones hay que tener muy claro que las variables utilizadas como parámetros (por valor) o las variables que se definen dentro de la función son locales a esa función, es decir, su ámbito es la función y fuera de ella esas variables no existen. Presta atención al programa `Varios.java` y observa que las dos funciones que hay definidas dentro de la clase utilizan sendas variables con nombre `x`. Pues bien, cada una de esas variables son completamente independientes, se trata de dos variables distintas que tienen el mismo nombre pero que son válidas cada una en su propio ámbito.

8.5 Paso de parámetros por valor y por referencia

En Java, como en la mayoría de lenguajes de programación existen dos maneras de pasar parámetros: por valor y por referencia.

Cuando se pasa un parámetro por valor, en realidad se pasa una copia de la variable, únicamente importa el valor. Cualquier modificación que se le haga a la variable que

se pasa como parámetro dentro de la función no tendrá ningún efecto fuera de la misma. Veamos un ejemplo.

```
/**
 * Paso de parámetros por valor
 *
 * @author Luis José Sánchez
 */
public class PruebaParametros1 {
  public static void main(String[] args) {

    int n = 10;

    System.out.println(n);
    calcula(n);
    System.out.println(n);
  }

  public static void calcula(int x) {
    x += 24;
    System.out.println(x);
  }
}
```

La salida del programa anterior es la siguiente:

```
10
34
10
```

A pesar de que la variable que se pasa como parámetro se modifica dentro de la función, los cambios no tienen ningún efecto en el programa principal.

Cuando se pasa un parámetro por referencia, por el contrario, si se modifica su valor dentro de la función, los cambios se mantienen una vez que la función ha terminado de ejecutarse.

En la mayoría de los lenguajes de programación es el programador quien decide cuándo un parámetro se pasa por valor y cuándo se pasa por referencia. En Java no podemos elegir. Todos los parámetros que son de tipo `int`, `double`, `float`, `char` o `String` se pasan siempre por valor mientras que los *arrays* se pasan siempre por referencia. Esto quiere decir que cualquier cambio que efectuemos en un array que se pasa como parámetro permanece cuando termina la ejecución de la función, por lo que hay que tener especial cuidado en estos casos.

```
/**
 * Paso de un array como parámetro
 *
 * @author Luis José Sánchez
 */
public class PruebaParametrosArray {
  public static void main(String[] args) {

    int n[] = {8, 33, 200, 150, 11};
    int m[] = new int[5];

    muestraArray(n);
    incrementa(n);
    muestraArray(n);
  }

  public static void muestraArray(int x[]) {
    for (int i = 0; i < x.length; i++) {
      System.out.print(x[i] + " ");
    }
    System.out.println();
  }

  public static void incrementa(int x[]) {
    for (int i = 0; i < x.length; i++) {
      x[i]++;
    }
  }
}
```

La salida del programa anterior es la siguiente:

```
8 33 200 150 11
9 34 201 151 12
```

Comprobamos, por tanto, que el array que se ha pasado como parámetro se ha modificado en la función y se han conservado los cambios en el programa principal.

Por último, se muestra un programa que contiene funciones que operan con arrays de dos dimensiones.

```
import matematicas.Varias;

/**
 * Paso de un array bidimensional como parámetro
 *
 * @author Luis José Sánchez
 */
public class ArrayBiFunciones {
  public static void main(String[] args) {

    int[][] n = new int[6][9];

    for(int i = 0; i < 6; i++) {
      for(int j = 0; j < 9; j++) {
        n[i][j] = (int)(Math.random()*100000);
      }
    }

    muestraArrayIntBi(n);
  }


  // Funciones ///////////////////////////////////////////////////////

  /**
   * Devuelve el número de filas de un array bidimensional de
   * números enteros.
   *
   * @param x un array bidimensional de números enteros
   * @return  número de filas del array
   */
  public static int filasArrayIntBi(int x[][]) {
    return x.length;
  }


  /**
   * Devuelve el número de columnas de un array bidimensional
   * de números enteros.
   *
   * @param x un array bidimensional de números enteros
   * @return  número de columnas del array
   */
  public static int columnasArrayIntBi(int x[][]) {
    return x[0].length;
  }
```

```
  /**
   * Devuelve el máximo de un array bidimensional
   * de números enteros.
   *
   * @param x un array bidimensional de números enteros
   * @return  el valor máximo encontrado en el array
   */
  public static int maximoArrayIntBi(int x[][]) {

    int maximo = Integer.MIN_VALUE;

    for (int f = 0; f < filasArrayIntBi(x); f++) {
      for (int c = 0; c < columnasArrayIntBi(x); c++) {
        if (x[f][c] > maximo) {
          maximo = x[f][c];
        }
      }
    }

    return maximo;
  }


  /**
   * Muestra por pantalla el contenido de un array bidimensional
   * de números enteros.
   *
   * @param x un array bidimensional de números enteros
   */
  public static void muestraArrayIntBi(int x[][]) {

    String formatoNumero = "%" + matematicas.Varias.digitos(maximoArrayIntBi(x)) + "d";

    for (int f = 0; f < filasArrayIntBi(x); f++) {
      for (int c = 0; c < columnasArrayIntBi(x); c++) {
        System.out.printf(formatoNumero + " ", x[f][c]);
      }
      System.out.println();
    }
  }
}
```

8.6 Ejercicios



Ejercicios 1-14

Crea una biblioteca de funciones matemáticas que contenga las siguientes funciones. Recuerda que puedes usar unas dentro de otras si es necesario. Observa bien lo que hace cada función ya que, si las implementas en el orden adecuado, te puedes ahorrar mucho trabajo. Por ejemplo, la función `esCapicua` resulta trivial teniendo `voltea` y la función `siguientePrimo` también es muy fácil de implementar teniendo `esPrimo`.

1. **esCapicua**: Devuelve verdadero si el número que se pasa como parámetro es capicúa y falso en caso contrario.
2. **esPrimo**: Devuelve verdadero si el número que se pasa como parámetro es primo y falso en caso contrario.
3. **siguientePrimo**: Devuelve el menor primo que es mayor al número que se pasa como parámetro.
4. **potencia**: Dada una base y un exponente devuelve la potencia.
5. **digitos**: Cuenta el número de dígitos de un número entero.
6. **voltea**: Le da la vuelta a un número.
7. **digitoN**: Devuelve el dígito que está en la posición n de un número entero. Se empieza contando por el 0 y de izquierda a derecha.
8. **posicionDeDigito**: Da la posición de la primera ocurrencia de un dígito dentro de un número entero. Si no se encuentra, devuelve -1.
9. **quitaPorDetras**: Le quita a un número n dígitos por detrás (por la derecha).
10. **quitaPorDelante**: Le quita a un número n dígitos por delante (por la izquierda).
11. **pegaPorDetras**: Añade un dígito a un número por detrás.
12. **pegaPorDelante**: Añade un dígito a un número por delante.
13. **trozoDeNumero**: Toma como parámetros las posiciones inicial y final dentro de un número y devuelve el trozo correspondiente.
14. **juntaNumeros**: Pega dos números para formar uno.



Ejercicio 15

Muestra los números primos que hay entre 1 y 1000.



Ejercicio 16

Muestra los números capicúa que hay entre 1 y 99999.



Ejercicio 17

Escribe un programa que pase de binario a decimal.



Ejercicio 18

Escribe un programa que pase de decimal a binario.



Ejercicio 19

Une y amplía los dos programas anteriores de tal forma que se permita convertir un número entre cualquiera de las siguientes bases: decimal, binario, hexadecimal y octal.



Ejercicios 20-28

Crea una biblioteca de funciones para arrays (de una dimensión) de números enteros que contenga las siguientes funciones:

1. **generaArrayInt**: Genera un array de tamaño n con números aleatorios cuyo intervalo (mínimo y máximo) se indica como parámetro.
2. **minimoArrayInt**: Devuelve el mínimo del array que se pasa como parámetro.
3. **maximoArrayInt**: Devuelve el máximo del array que se pasa como parámetro.
4. **mediaArrayInt**: Devuelve la media del array que se pasa como parámetro.
5. **estaEnArrayInt**: Dice si un número está o no dentro de un array.
6. **posicionEnArray**: Busca un número en un array y devuelve la posición (el índice) en la que se encuentra.
7. **volteaArrayInt**: Le da la vuelta a un array.
8. **rotaDerechaArrayInt**: Rota n posiciones a la derecha los números de un array.
9. **rotaIzquierdaArrayInt**: Rota n posiciones a la izquierda los números de un array.



Ejercicio 29-34

Crea una biblioteca de funciones para arrays bidimensionales (de dos dimensiones) de números enteros que contenga las siguientes funciones:

1. **generaArrayBiInt**: Genera un array de tamaño n x m con números aleatorios cuyo intervalo (mínimo y máximo) se indica como parámetro.
2. **filaDeArrayBiInt**: Devuelve la fila i-ésima del array que se pasa como parámetro.
3. **columnaDeArrayBiInt**: Devuelve la columna j-ésima del array que se pasa como parámetro.
4. **coordenadasEnArrayBiInt**: Devuelve la fila y la columna (en un array con dos elementos) de la primera ocurrencia de un número dentro de un array bidimensional. Si el número no se encuentra en el array, la función devuelve el array {-1, -1}.
5. **esPuntoDeSilla**: Dice si un número es o no punto de silla, es decir, mínimo en su fila y máximo en su columna.
6. **diagonal**: Devuelve un array que contiene una de las diagonales del array bidimensional que se pasa como parámetro. Se pasan como parámetros fila, columna y dirección. La fila y la columna determinan el número que marcará las dos posibles diagonales dentro del array. La dirección es una cadena de caracteres que puede ser “nose” o “neso”. La cadena “nose” indica que se elige la diagonal que va del noroeste hacia el sureste, mientras que la cadena “neso” indica que se elige la diagonal que va del noreste hacia el suroeste.



Ejercicio 35

Crea una función con la siguiente cabecera:

```
public static String convierteEnPalotes(int n)
```

Esta función convierte el número n al sistema de palotes y lo devuelve en una cadena de caracteres. Por ejemplo, el 470213 en decimal es el | | | | - | | | | | | | | - - | | - | - | | | en el sistema de palotes. Utiliza esta función en un programa para comprobar que funciona bien. Desde la función no se debe mostrar nada por pantalla, solo se debe usar `print` desde el programa principal.



Ejercicio 36

Crea la función de manejo de arrays que tenga la siguiente cabecera y que haga lo que se especifica en los comentarios (puedes incluirla en tu propia biblioteca de rutinas):

```
public static int[] filtraPrimos(int x[]) // Devuelve un array con todos los
                                          // números primos que se encuentren
                                          // en otro array que se pasa como
                                          // parámetro.
                                          // Obviamente el tamaño del array
                                          // que se devuelve será menor o
                                          // igual al que se pasa como
                                          // parámetro.
```

Utiliza esta función en un programa para comprobar que funcionan bien. Para que el ejercicio resulte más fácil, las repeticiones de primos se conservan; es decir, si en el array x el número 13 se repite 3 veces, en el array devuelto también estará repetido 3 veces. Si no existe ningún número primo en x, se devuelve un array con el número -1 como único elemento.



Ejercicio 37

Crea una función con la siguiente cabecera:

```
public String convierteEnMorse(int n)
```

Esta función convierte el número n al sistema Morse y lo devuelve en una cadena de caracteres. Por ejemplo, el 213 es el `. . _ _ _ . _ _ _ _ . . . _ _` en Morse. Utiliza esta función en un programa para comprobar que funciona bien. Desde la función no se debe mostrar nada por pantalla, solo se debe usar print desde el programa principal.

```
1 . _ _ _ _     6 _ . . . .
2 . . _ _ _     7 _ _ . . .
3 . . . _ _     8 _ _ _ . .
4 . . . . _     9 _ _ _ _ .
5 . . . . .     0 _ _ _ _ _
```



Ejercicio 38

Crea la función de manejo de arrays que tenga la siguiente cabecera y que haga lo que se especifica en los comentarios (puedes incluirla en tu propia biblioteca de rutinas):

```
public int[] filtraCapicuas(int x[]) // Devuelve un array con todos los números
                                     // capicúa que se encuentren en otro array
                                     // que se pasa como parámetro.
                                     // Obviamente el tamaño del array que se
                                     // devuelve será menor o igual al que se
                                     // pasa como parámetro.
```

Utiliza esta función en un programa para comprobar que funcionan bien. Para que el ejercicio resulte más fácil, las repeticiones de números capicúa se conservan; es decir, si en el array x el número 505 se repite 3 veces, en el array devuelto también estará repetido 3 veces. Si no existe ningún número capicúa en x, se devuelve un array con el número -1 como único elemento.



Ejercicio 39

Crea una función con la siguiente cabecera:

```
public String convierteEnPalabras(int n)
```

Esta función convierte los dígitos del número n en las correspondientes palabras y lo devuelve todo en una cadena de caracteres. Por ejemplo, el 470213 convertido a palabras sería:

```
cuatro, siete, cero, dos, uno, tres
```

Utiliza esta función en un programa para comprobar que funciona bien. Desde la función no se debe mostrar nada por pantalla, solo se debe usar `print` desde el programa principal. Fíjate que hay una coma detrás de cada palabra salvo al final.



Ejercicio 40

Crea la función de manejo de arrays que tenga la siguiente cabecera y que haga lo que se especifica en los comentarios (puedes incluirla en tu propia biblioteca de rutinas):

```
public int[] filtraCon7(int x[]) // Devuelve un array con todos los números
                                 // que contienen el 7 (por ej. 7, 27, 782)
                                 // que se encuentren en otro array que se
                                 // pasa como parámetro. El tamaño del array
                                 // que se devuelve será menor o igual al
                                 // que se pasa como parámetro.
```

Utiliza esta función en un programa para comprobar que funcionan bien. Para que el ejercicio resulte más fácil, las repeticiones de números que contienen 7 se conservan; es decir, si en el array x el número 875 se repite 3 veces, en el array devuelto también estará repetido 3 veces. Si no existe ningún número que contiene 7 en el array x, se devuelve un array con el número -1 como único elemento.



Ejercicio 41

Realiza un programa que pinte un triángulo relleno tal como se muestra en los ejemplos. El usuario debe introducir la altura de la figura. Este ejercicio ya se realizó en el tema de bucles, ahora se trata de usar una función para que la implementación sea más sencilla. Por ejemplo, se puede crear una función `linea(char caracter, int repeticiones)` que pinte una línea con el carácter especificado.

Ejemplo 1:

```
Introduzca la altura de la figura: 8
********
*******
******
*****
****
***
**
*
```

Ejemplo 2:

```
Introduzca la altura de la figura: 5
*****
****
```

```
***
**
*
```



Ejercicio 42

Realiza un programa que pinte un triángulo hueco tal como se muestra en los ejemplos. El usuario debe introducir la altura de la figura. Utiliza funciones para pintar las líneas.

Ejemplo 1:

```
Introduzca la altura de la figura: 8
********
*     *
*    *
*   *
*  *
* *
**
*
```

Ejemplo 2:

```
Introduzca la altura de la figura: 5
*****
*  *
* *
**
*
```



Ejercicio 43

Realiza un programa que pinte un triángulo relleno tal como se muestra en los ejemplos. El usuario debe introducir la altura de la figura. Utiliza funciones para pintar las líneas.

Ejemplo 1:

```
Introduzca la altura de la figura: 8
********
 *******
  ******
```

```
   *****
    ****
     ***
      **
       *
```

Ejemplo 2:

```
Introduzca la altura de la figura: 5
*****
 ****
  ***
   **
    *
```



Ejercicio 44

Realiza un programa que pinte un triángulo hueco tal como se muestra en los ejemplos. El usuario debe introducir la altura de la figura. Utiliza funciones para pintar las líneas.

Ejemplo 1:

```
Introduzca la altura de la figura: 8
********
 *     *
  *    *
   *   *
    *  *
     * *
      **
       *
```

Ejemplo 2:

```
Introduzca la altura de la figura: 5
*****
 *  *
  * *
   **
    *
```



Ejercicio 45

Realiza un programa que pinte un valle tal como se muestra en los ejemplos. El usuario debe introducir la altura de la figura. Podemos suponer que el usuario introduce una altura mayor o igual a 3.

Ejemplo 1:

```
Introduzca la altura de la figura: 8

       *             *
       **           **
       ***         ***
       ****       ****
       *****     *****
       ******   ******
       ******* *******
       ***************
```

Ejemplo 2:

```
Introduzca la altura de la figura: 5

       *       *
       **     **
       ***   ***
       **** ****
       *********
```

Ejemplo 3:

```
Introduzca la altura de la figura: 3

       *   *
       ** **
       *****
```



Ejercicio 46

Realiza un programa que pinte un valle tal como se muestra en los ejemplos. El usuario debe introducir la altura de la figura. Podemos suponer que el usuario introduce una altura mayor o igual a 3.

Ejemplo 1:

```
Introduzca la altura de la figura: 8

       *             *
       **           **
       * *         * *
       *  *       *  *
       *   *     *   *
```

```
*    *   *    *
*     * *     *
***************
```

Ejemplo 2:

```
Introduzca la altura de la figura: 5

*       *
**     **
* *   * *
*  * *  *
*********
```

Ejemplo 3:

```
Introduzca la altura de la figura: 3

*   *
** **
*****
```



Ejercicio 47

Define la función **convierteArrayEnString** con la siguiente cabecera:

```
public static String convierteArrayEnString(int[] a)
```

Esta función toma como parámetro un array que contiene números y devuelve una cadena de caracteres con esos números. Por ejemplo, si a = { }, convierteArrayEnString(a) devuelve “”; si a = { 8 }, convierteArrayEnString(a) devuelve “8”; si a = { 6, 2, 5, 0, 1 }, convierteArrayEnString(a) devuelve “62501”.



Ejercicio 48

Define la función **concatena** con la siguiente cabecera:

```
public static int[] concatena(int[] a, int[] b)
```

Esta función toma dos arrays como parámetros y devuelve un array que es el resultado de concatenar ambos. Por ej. si a = { 8, 9, 0 } y b = { 1, 2, 3 }, concatena(a, b) devuelve { 8, 9, 0, 1, 2, 3 }.



Ejercicio 49

Escribe un programa que genere los n primeros términos de la sucesión *look and say*. El primer término es 1. A continuación se va leyendo - un uno - por tanto tenemos 11, se sigue leyendo – dos unos – por tanto tenemos 21, etc. Se recomienda usar arrays para almacenar los dígitos porque los tipos `int` y `long` son muy limitados en cuanto al número de dígitos. También puede resultar de ayuda utilizar las funciones `convierteArrayEnString` y `concatena` definidas en los ejercicios anteriores.

Ejemplo 1:

```
¿Cuántos términos de la sucesión look and say quiere calcular? 8
1, 11, 21, 1211, 111221, 312211, 13112221, 1113213211
```

Ejemplo 2:

```
¿Cuántos términos de la sucesión look and say quiere calcular? 5
1, 11, 21, 1211, 111221
```

Ejemplo 3:

```
¿Cuántos términos de la sucesión look and say quiere calcular? 12
1, 11, 21, 1211, 111221, 312211, 13112221, 1113213211, 31131211131221, 13211311123113112211, 1\
1131221133112132113212221, 3113112221232112111312211312113211
```



Ejercicio 50

Define la función **mezcla** con la siguiente cabecera:

```
public static int[] mezcla(int[] a, int[] b)
```

Esta función toma dos arrays como parámetros y devuelve un array que es el resultado de mezclar los números de ambos de forma alterna, se coge un número de a, luego de b, luego de a, etc. Los arrays a y b pueden tener longitudes diferentes; por tanto, si se terminan los números de un array se terminan de coger todos los que quedan del otro.

Ejemplos:

```
Si a = {8, 9, 0} y b = {1, 2, 3}, mezcla(a, b) devuelve {8, 1, 9, 2, 0, 3 }
Si a = {4, 3} y b = {7, 8, 9, 10}, mezcla(a, b) devuelve {4, 7, 3, 8, 9, 10}
Si a = {8, 9, 0, 3} y b = {1}, mezcla(a, b) devuelve {8, 1, 9, 0, 3}
Si a = { } y b = {1, 2, 3}, mezcla(a, b) devuelve {1, 2, 3}
```



Ejercicio 51

Realiza un programa que rellene un array con 10 números aleatorios comprendidos entre 2 y 100 (ambos incluidos) y que los muestre por pantalla. A continuación, el programa indicará para cada uno de ellos si es un número primo y/o un capicúa de la forma que muestra el ejemplo.

Ejemplos:

```
Array generado:
19 22 57 11 3 52 32 46 2 14
El 19 es primo y no es capicúa.
El 22 no es primo y es capicúa.
El 57 no es primo y no es capicúa.
El 11 es primo y es capicúa.
El 3 es primo y es capicúa.
El 52 no es primo y no es capicúa.
El 32 no es primo y no es capicúa.
El 46 no es primo y no es capicúa.
El 2 es primo y es capicúa.
14 no es primo y no es capicúa.
```



Ejercicio 52

Implementa la función **aleatorioDeArray** con la cabecera que se muestra a continuación:

```
public static int aleatorioDeArray(int[] a)
```

Esta función debe devolver un número del array escogido al azar entre todos los disponibles. Por ejemplo, si a = {111, 222, 333, 444}, aleatorioDeArray(a) podría devolver el 111, el 222, el 333 o el 444. Si b = {52, 37}, aleatorioDeArray(b) podría devolver el 52 o el 37. Utiliza la función en un programa de prueba.



Ejercicio 53

Implementa una función con nombre **nEsimo** que busque el número que hay dentro de un array bidimensional en la posición n-ésima contando de izquierda a derecha y de arriba abajo, como si se estuviera leyendo. El primer elemento es el 0. Si la posición donde se busca no existe en el array, la función debe devolver -1. Se debe entregar tanto el código de la función como el código de prueba que la usa. La cabecera de la función es la siguiente:

```
public static int nEsimo(int[][] n, int posicion)
```

Si el array a es el que se muestra a continuación:

```
35 72 24 45 42 60
32 42 64 23 41 39
98 45 94 11 18 48
12 34 56 78 90 12

nEsimo(a, 0) devuelve 35
nEsimo(a, 2) devuelve 24
nEsimo(a, 5) devuelve 60
nEsimo(a, 6) devuelve 32
nEsimo(a, 21) devuelve 78
nEsimo(a, 23) devuelve 12
nEsimo(a, 24) devuelve -1
nEsimo(a, 100) devuelve -1
```



Ejercicio 54

Crea las funciones cuyas cabeceras se muestran a continuación, observa que tienen el mismo nombre:

```
public static int ocurrencias(int digito, int n)
public static int ocurrencias(int digito, int[] a)
```

La función **ocurrencias** devuelve el número de veces que aparece un dígito dentro de un número (primera función) o bien el número de veces que aparece un dígito en una serie de números contenidos en un array (segunda función).

Ejemplos:

`console ocurrencias(8, 4672) devuelve 0 ocurrencias(5, 25153) devuelve 2 ocurrencias(2, 123456) devuelve 1 Si a = {714, 81, 9, 11}, ocurrencias(1, a) devuelve 4 Si a = {42, 13, 12345, 4}, ocurrencias(4, a) devuelve 3 Si a = {6, 66, 666}, ocurrencias(6, a) devuelve 6` console

Utiliza estas funciones en un programa para comprobar que funcionan bien.



Ejercicio 55

Realiza una función que tome como parámetro un array de cadenas de caracteres y que devuelva otro array con los mismos valores habiendo eliminado las posibles repeticiones. Se distinguen mayúsculas de minúsculas, por tanto "hola" es distinto de "Hola". Por ejemplo, si el array **a** contiene los valores **{"casa", "coche", "sol", "mesa", "mesa", "coche", "ordenador", "sol", "CASA"}**, la sentencia **sinRepetir(a)** devolvería el array **{"casa", "coche", "sol", "mesa", "ordenador", "CASA"}**. Se debe entregar tanto el código de la función como el código de prueba que la usa. La cabecera de la función es la siguiente:

```
public static String[] sinRepetir(String[] s)
```



Ejercicio 56

Implementa una función con nombre **corteza** que sea capaz de extraer la capa exterior de un array bidimensional. Esta capa se extrae en forma de array de una dimensión. La extracción de números comienza en la esquina superior izquierda y continúa en el sentido de las agujas del reloj. Se debe entregar tanto el código de la función como el código de prueba que la usa. La cabecera de la función es la siguiente:

```
public static int[] corteza(int[][] n)
```

Por ejemplo, si el array bidimensional **a** es el que se muestra a continuación:

```
45 92 14 20 25 78
35 72 24 45 42 60
32 42 64 23 41 39
98 45 94 11 18 48
```

El array unidimensional generado por **corteza(a)** sería el siguiente:

```
45 92 14 20 25 78 60 39 48 18 11 94 45 98 32 35
```

9. Programación orientada a objetos

9.1 Clases y objetos

La programación orientada a objetos es un paradigma de programación que se basa, como su nombre indica, en la utilización de objetos. Estos objetos también se suelen llamar instancias.

Un objeto en términos de POO no se diferencia mucho de lo que conocemos como un objeto en la vida real. Pensemos por ejemplo en un coche. Nuestro coche sería un objeto concreto de la vida real, igual que el coche del vecino, o el coche de un compañero de trabajo, o un deportivo que vimos por la calle el fin de semana pasado… Todos esos coches son objetos concretos que podemos ver y tocar.

Tanto mi coche como el coche del vecino tienen algo en común, ambos son coches. En este caso mi coche y el coche del vecino serían **instancias** (objetos) y coche (a secas) sería una **clase**. La palabra coche define algo genérico, es una abstracción, no es un coche concreto sino que hace referencia a unos elementos que tienen una serie de propiedades como matrícula, marca, modelo, color, etc.; este conjunto de propiedades se denominan **atributos** o **variables de instancia**.



Clase

Concepto abstracto que denota una serie de cualidades, por ejemplo **coche**.

Instancia

Objeto palpable, que se deriva de la concreción de una clase, por ejemplo **mi coche**.

Atributos

Conjunto de características que comparten los objetos de una clase, por ejemplo para la clase **coche** tendríamos **matrícula**, **marca**, **modelo**, **color** y **número de plazas**.

En Java, los nombres de las clases se escriben con la primera letra en mayúscula mientras que los nombres de las instancias comienzan con una letra en minúscula. Por ejemplo, la clase coche se escribe en Java como `Coche` y el objeto "mi coche" se podría escribir como `miCoche`.

Definiremos cada clase en un fichero con el mismo nombre más la extensión `.java`, por tanto, la definición de la clase `Coche` debe estar contenida en un fichero con nombre `Coche.java`

Vamos a definir a continuación la clase `Libro` con los atributos `isbn`, `autor`, `titulo` y `numeroPaginas`.

```
/**
 * Libro.java
 * Definición de la clase Libro
 * @author Luis José Sánchez
 */
public class Libro {
  // atributos
  String isbn;
  String titulo;
  String autor;
  int numeroDePaginas;
}
```

A continuación creamos varios objetos de esta clase.

```
/**
 * PruebaLibro.java
 * Programa que prueba la clase Libro
 * @author Luis José Sánchez
 */
public class PruebaLibro {
  public static void main(String[] args) {
    Libro lib = new Libro();
    Libro miLibrito = new Libro();
    Libro quijote = new Libro();
  }
}
```

Hemos creado tres instancias de la clase libro: `lib`, `miLibrito` y `quijote`. Ya sabemos definir una clase indicando sus atributos y crear instancias.



Las **variables de instancia** (atributos) determinan las **cualidades** de los objetos.

9.2 Encapsulamiento y ocultación

Uno de los pilares en los que se basa la Programación Orientada a Objetos es el **encapsulamiento**. Básicamente, el **encapsulamiento** consiste en definir todas las

propiedades y el comportamiento de una clase dentro de esa clase; es decir, en la clase `Coche` estará definido todo lo concerniente a la clase `Coche` y en la clase `Libro` estará definido todo lo que tenga que ver con la clase `Libro`.

El **encapsulamiento** parece algo obvio, casi de perogrullo, pero hay que tenerlo siempre muy presente al programar utilizando clases y objetos. En alguna ocasión puede que estemos tentados a mezclar parte de una clase con otra clase distinta para resolver un problema puntual. No hay que caer en esa trampa. Se deben escribir los programas de forma que cada cosa esté en su sitio. Sobre todo al principio, cuando definimos nuestras primeras clases, debemos estar pendientes de que todo está definido donde corresponde.

La **ocultación** es una técnica que incorporan algunos lenguajes (entre ellos Java) que permite esconder los elementos que definen una clase, de tal forma que desde otra clase distinta no se pueden "ver las tripas" de la primera. La **ocultación** facilita, como veremos más adelante, el **encapsulamiento**.

9.3 Métodos

Un coche arranca, para, se aparca, hace sonar el claxon, se puede llevar a reparar… Un gato puede comer, dormir, maullar, ronronear…

Las acciones asociadas a una clase se llaman métodos. Estos métodos se definen dentro del cuerpo de la clase y se suelen colocar a continuación de los atributos.



Los **métodos** determinan el **comportamiento** de los objetos.

Creí haber visto un lindo gatito

Vamos a crear la clase `GatoSimple`. La llamamos así porque más adelante crearemos otra clase algo más elaborada que se llamará `Gato`. Para saber qué atributos debe tener esta clase hay que preguntarse qué características tienen los gatos. Todos los gatos son de un color determinado, pertenecen a una raza, tienen una edad, tienen un determinado sexo - son machos o hembras - y tienen un peso que se puede expresar en kilogramos. Éstos serán por tanto los atributos que tendrá la clase `GatoSimple`.

Para saber qué metodos debemos implementar hay que preguntarse qué acciones están asociadas a los gatos. Bien, pues los gatos maullan, ronronean, comen y si son machos se pelean entre ellos para disputarse el favor de las hembras. Esos serán los métodos que definamos en la clase.

```java
/**
 * GatoSimple.java
 * Definición de la clase GatoSimple
 * @author Luis José Sánchez
 */
public class GatoSimple {

  // atributos /////////////////////////////

  String color, raza, sexo;
  int    edad;
  double peso;

  // métodos ///////////////////////////////

  // constructor
  GatoSimple (String s) {
    this.sexo = s;
  }

  // getter
  String getSexo() {
    return this.sexo;
  }

  /**
   * Hace que el gato maulle
   */
  void maulla() {
    System.out.println("Miauuuu");
  }

  /**
   * Hace que el gato ronronee
   */
  void ronronea() {
    System.out.println("mrrrrrr");
  }

  /**
   * Hace que el gato coma.
   * A los gatos les gusta el pescado, si le damos otra comida
   * la rechazará.
   *
   * @param comida la comida que se le ofrece al gato
   */
```

```
  void come(String comida) {
    if (comida.equals("pescado")) {
        System.out.println("Hmmmm, gracias");
    } else {
        System.out.println("Lo siento, yo solo como pescado");
    }
  }

  /**
   * Pone a pelear dos gatos.
   * Solo se van a pelear dos machos entre sí.
   *
   * @param contrincante es el gato contra el que pelear
   */
  void peleaCon(GatoSimple contrincante) {
    if (this.sexo.equals("hembra")) {
      System.out.println("no me gusta pelear");
    } else {
      if (contrincante.getSexo().equals("hembra")) {
          System.out.println("no peleo contra gatitas");
      } else {
          System.out.println("ven aquí que te vas a enterar");
      }
    }
  }
}
```



Constructor

El método constructor tiene siempre el mismo nombre que la clase y se utiliza normalmente para inicializar los atributos.

Los atributos de la clase `GatoSimple` - `color`, `raza`, `sexo`, `edad` y `peso` - se declaran igual que las variables que hemos venido usando hasta ahora, pero hay una gran diferencia entre estos atributos y las variables que aparecen en el `main` (programa principal). Una variable definida en el cuerpo del programa principal es única, sin embargo cada uno de los objetos que se crean en el programa principal tienen sus propios atributos; es decir, si en el programa principal se crean 20 objetos de la clase `GatoSimple`, cada uno tiene sus valores para los atributos `color`, `raza`, etc.

Fíjate en la cabecera del método `peleaCon`:

```
void peleaCon(GatoSimple contrincante)
```

Como puedes comprobar, es posible pasar un objeto como parámetro. A continuación

se muestra un programa que prueba la clase `GatoSimple`. Te recomiendo seguir el programa línea a línea y observar atentamente la salida que produce.

```
/**
 * PruebaGatoSimple.java
 * Programa que prueba la clase GatoSimple
 * @author Luis José Sánchez
 */
public class PruebaGatoSimple {
  public static void main(String[] args) {

    GatoSimple garfield = new GatoSimple("macho");

    System.out.println("hola gatito");
    garfield.maulla();
    System.out.println("toma tarta");
    garfield.come("tarta selva negra");
    System.out.println("toma pescado, a ver si esto te gusta");
    garfield.come("pescado");

    GatoSimple tom = new GatoSimple("macho");

    System.out.println("Tom, toma sopita de verduras");
    tom.come("sopa de verduras");

    GatoSimple lisa = new GatoSimple("hembra");

    System.out.println("gatitos, a ver cómo maulláis");
    garfield.maulla();
    tom.maulla();
    lisa.maulla();

    garfield.peleaCon(lisa);
    lisa.peleaCon(tom);
    tom.peleaCon(garfield);
  }
}
```

Observa cómo al crear una instancia se llama al constructor que, como decíamos antes, tiene el mismo nombre de la clase y sirve para inicializar los atributos. En este caso se inicializa el atributo `sexo`. Más adelante veremos constructores que inicializan varios atributos e incluso definiremos distintos constructores en la misma clase.

```
GatoSimple garfield = new GatoSimple("macho");
```

Como puedes ver hay métodos que no toman ningún parámetro.

```
garfield.maulla();
```

Y hay otros métodos que deben tomar parámetros obligatoriamente.

```
garfield.come("tarta selva negra");
```

Métodos *getter* y *setter*

Vamos a crear la clase `Cubo`. Para saber qué atributos se deben definir, nos preguntamos qué características tienen los cubos - igual que hicimos con la clase `GatoSimple`. Todos los cubos tienen una determinada capacidad, un color, están hechos de un determinado material - plástico, latón, etc. - y puede que tengan asa o puede que no. Un cubo se fabrica con el propósito de contener líquido; por tanto otra característica es la cantidad de litros de líquido que contiene en un momento determinado. Por ahora, solo nos interesa saber la capacidad máxima y los litros que contiene el cubo en cada momento, así que esos serán los atributos que tendremos en cuenta.

```
/**
 * Cubo.java
 * Definición de la clase Cubo
 * @author Luis José Sánchez
 */
public class Cubo {

  // atributos ////////////////////////////

  int capacidad; // capacidad máxima en litros
  int contenido; // contenido actual en litros

  // métodos //////////////////////////////

  // constructor
  Cubo (int c) {
    this.capacidad = c;
  }

  // métodos getter
  int getCapacidad() {
    return this.capacidad;
  }

  int getContenido() {
    return this.contenido;
  }
```

```
// método setter
void setContenido(int litros) {
  this.contenido = litros;
}

// otros métodos
void vacia() {
  this.contenido = 0;
}

/**
 * Llena el cubo al máximo de su capacidad.
 */
void llena() {
  this.contenido = this.capacidad;
}

/**
 * Pinta el cubo en la pantalla.
 * Se muestran los bordes del cubo con el carácter # y el
 * agua que contiene con el carácter ~.
 */
void pinta() {
  for (int nivel = this.capacidad; nivel > 0; nivel--) {
    if (this.contenido >= nivel) {
      System.out.println("#~~~~#");
    } else {
      System.out.println("#    #");
    }
  }
  System.out.println("######");
}

/**
 * Vuelca el contenido de un cubo sobre otro.
 * Antes de echar el agua se comprueba cuánto le cabe al
 * cubo destino.
 */
void vuelcaEn(Cubo destino) {
  int libres = destino.getCapacidad() - destino.getContenido();

  if (libres > 0) {
    if (this.contenido <= libres) {
      destino.setContenido(destino.getContenido() + this.contenido);
      this.vacia();
```

```
      } else {
        this.contenido -= libres;
        destino.llena();
      }
    }
  }
}
```

Observa estos métodos extraídos de la clase `Cubo`:

```
int getCapacidad() {
    return this.capacidad;
}

int getContenido() {
    return this.contenido;
}
```

Se trata de métodos muy simples, su cometido es devolver el valor de un atributo. Podrían tener cualquier nombre pero en Java es costumbre llamarlos con la palabra *get* (obtener) seguida del nombre del atributo.

Fijémonos ahora este otro método:

```
void setContenido(int litros) {
    this.contenido = litros;
}
```

Ahora estamos ante un *setter*. Este tipo de métodos tiene el cometido de establecer un valor para un determinado atributo. Como puedes ver, `setContenido` está formada por *set* (asignar) más el nombre del atributo.

Podrías preguntarte: ¿por qué se crea un método *getter* para extraer el valor de una variable y no se accede a la variable directamente?

Parece más lógico hacer esto

```
System.out.print(miCubo.capacidad);
```

que esto otro

```
System.out.print(miCubo.getCapacidad());
```

Sin embargo, en Java se opta casi siempre por esta última opción. Tiene su explicación y lo entenderás bien cuando estudies el apartado Ámbito/visibilidad de los elementos de una clase - public , protected y private

Probamos, con el siguiente ejemplo, la clase `Cubo` que acabamos de definir. Crea tus propios cubos, pasa agua de unos a otros y observa lo que se muestra por pantalla.

```java
/**
 * PruebaCubo.java
 * Programa que prueba la clase Cubo
 * @author Luis José Sánchez
 */
public class PruebaCubo {
  public static void main(String[] args) {

    Cubo cubito = new Cubo(2);
    Cubo cubote = new Cubo(7);

    System.out.println("Cubito: \n");
    cubito.pinta();

    System.out.println("\nCubote: \n");
    cubote.pinta();

    System.out.println("\nLleno el cubito: \n");
    cubito.llena();
    cubito.pinta();

    System.out.println("\nEl cubote sigue vacío: \n");
    cubote.pinta();

    System.out.println("\nAhora vuelco lo que tiene el cubito en el cubote.\n");
    cubito.vuelcaEn(cubote);

    System.out.println("Cubito: \n");
    cubito.pinta();

    System.out.println("\nCubote: \n");
    cubote.pinta();

    System.out.println("\nAhora vuelco lo que tiene el cubote en el cubito.\n");
    cubote.vuelcaEn(cubito);

    System.out.println("Cubito: \n");
    cubito.pinta();

    System.out.println("\nCubote: \n");
    cubote.pinta();
  }
}
```

Método `toString`

La definición de la clase `Cubo` del apartado anterior contiene el método `pinta` que, como su nombre indica, permite pintar en pantalla un objeto de esa clase. Se podría definir el método `ficha` para mostrar información sobre objetos de la clase `Alumno` por ejemplo. También sería posible implementar el método `imprime` dentro de la clase Libro con el propósito de mostrar por pantalla los datos de un libro. Todos estos métodos - pinta, ficha e imprime - hacen básicamente lo mismo.

En Java existe una solución muy elegante para mostrar información sobre un objeto por pantalla. Si se quiere mostrar el contenido de la variable entera `x` se utiliza `System.out.print(x)` y si se quiere mostrar el valor de la variable de tipo cadena de caracteres `nombre` se escribe `System.out.print(nombre)`. De la misma manera, si se quiere mostrar el objeto `miPiramide` que pertenece a la clase `Piramide`, también se podría usar `System.out.print(miPiramide)`. Java sabe perfectamente cómo mostrar números y cadenas de caracteres pero no sabe *a priori* cómo se pintan pirámides. Para indicar a Java cómo debe pintar un objeto de la clase `Piramide` basta con implementar el método `toString` dentro de la clase.

Veamos un ejemplo muy sencillo de implementación de `toString`. Definiremos la clase `Cuadrado` con el atributo `lado`, el constructor y el método `toString`.

```
/**
 * Cuadrado.java
 * Definición de la clase Cuadrado
 * @author Luis José Sánchez
 */
public class Cuadrado {

  int lado;

  public Cuadrado(int l) {
    this.lado = l;
  }

  public String toString() {

    int i, espacios;
    String resultado = "";

    for (i = 0; i < this.lado; i++) {
      resultado += "□□";
    }
    resultado += "\n";

    for (i = 1; i < this.lado - 1; i++) {
      resultado += "□□";
```

```
      for (espacios = 1; espacios < this.lado - 1; espacios++) {
        resultado += "  ";
      }
      resultado += "□□\n";
    }

    for (i = 0; i < this.lado; i++) {
      resultado += "□□";
    }
    resultado += "\n";

    return resultado;
  }
}
```

Observa que el método `toString()` devuelve una cadena de caracteres. Esa cadena es precisamente la que mostrará por pantalla `System.out.print()`. En el programa que se muestra a continuación y que prueba la clase `Cuadrado` puedes comprobar que se pinta un cuadrado igual que si se tratara de cualquier otra variable.

```
/**
 * PruebaCuadrado.java
 * Programa que prueba la clase Cuadrado
 * @author Luis José Sánchez
 */
public class PruebaCuadrado {
  public static void main(String[] args) {

    Cuadrado miCuadradito = new Cuadrado(5);
    System.out.println(miCuadradito);
  }
}
```

9.4 Ámbito/visibilidad de los elementos de una clase - `public`, `protected` y `private`

Al definir los elementos de una clase, se pueden especificar sus ámbitos (*scope*) de visibilidad o accesibilidad con las palabras reservadas `public` (público), `protected` (protegido) y `private` (privado).

En la siguiente tabla se muestra desde dónde es visible/accesible un elemento (atributo o método) según el modificador que lleve.

	En la misma clase	En el mismo paquete	En una subclase	Fuera del paquete
private	✔	✖	✖	✖
protected	✔	✔	✔	✖
public	✔	✔	✔	✔
sin especificar	✔	✔	✖	✖

Figura 9.1: Ámbito de los elementos de una clase.

Por ejemplo, un atributo `private` solamente es accesible desde la misma clase, sin embargo, a un método `protected` se podrá acceder desde la misma clase donde esté definido, desde otro fichero dentro del mismo paquete o desde una subclase.

Como regla general, se suelen declarar `private` los atributos o variables de instancia y `public` los métodos.

En el siguiente apartado veremos qué son las subclases. De momento fíjate en el ámbito del atributo y de los métodos en la definición de la clase `Animal`.

```java
/**
 * Animal.java
 * Definición de la clase Animal
 * @author Luis José Sánchez
 */
public abstract class Animal {

  private Sexo sexo;

  public Animal () {
    sexo = Sexo.MACHO;
  }

  public Animal (Sexo s) {
    sexo = s;
```

```
  }

  public Sexo getSexo() {
    return sexo;
  }

  public String toString() {
    return "Sexo: " + this.sexo + "\n";
  }

  /**
   * Hace que el animal se eche a dormir.
   */
  public void duerme() {
    System.out.println("Zzzzzzz");
  }
}
```

Fíjate que el atributo `sexo` se ha definido como `private`.

```
private Sexo sexo;
```

Eso quiere decir que a ese atributo únicamente se tiene acceso dentro de la clase `Animal`. Sin embargo, todos los métodos se han definido `public`, lo que significa que se podrán utilizar desde cualquier otro programa, por ejemplo, como veremos más adelante, desde el programa `PurebaAnimal.java`.



Salvo casos puntuales se seguirá la regla de declarar `private` las variables de instancia y `public` los métodos.

El sexo de un animal solo puede ser macho, hembra o hermafrodita. Una forma de delimitar los valores que puede tomar un atributo es definir un tipo enumerado.



Tipo enumerado

Mediante `enum` se puede definir un tipo enumerado, de esta forma un atributo solo podrá tener uno de los posibles valores que se dan como opción. Los valores que se especifican en el tipo enumerado se suelen escribir con todas las letras en mayúscula.

```java
/**
 * Sexo.java
 * Definición del tipo enumerado Sexo
 * @author Luis José Sánchez
 */
public enum Sexo {
    MACHO, HEMBRA, HERMAFRODITA
}
```

9.5 Herencia

La herencia es una de las características más importantes de la POO. Si definimos una serie de atributos y métodos para una clase, al crear una subclase, todos estos atributos y métodos siguen siendo válidos.

En el apartado anterior se define la clase `Animal`. Uno de los métodos de esta clase es `duerme`. A continuación podemos crear las clases `Gato` y `Perro` como subclases de `Animal`. De forma automática, se puede utilizar el método `duerme` con las instancias de las clases `Gato` y `Perro` ¿no es fantástico?

La clase `Ave` es subclase de `Animal` y la clase `Pinguino`, a su vez, sería subclase de `Ave` y por tanto hereda todos sus atributos y métodos.



Clase abstracta (`abstract`)

Una clase abstracta es aquella que no va a tener instancias de forma directa, aunque sí habrá instancias de las subclases (siempre que esas subclases no sean también abstractas). Por ejemplo, si se define la clase `Animal` como abstracta, no se podrán crear objetos de la clase `Animal`, es decir, no se podrá hacer `Animal mascota = new Animal()`, pero sí se podrán crear instancias de la clase `Gato`, `Ave` o `Pinguino` que son subclases de `Animal`.

Para crear en Java una subclase de otra clase existente se utiliza la palabra reservada `extends`. A continuación se muestra el código de las clases `Gato`, `Ave` y `Pinguino`, así como el programa que prueba estas clases creando instancias y aplicándoles métodos. Recuerda que la definición de la clase `Animal` se muestra en el apartado anterior.

```
/**
 * Gato.java
 * Definición de la clase Gato
 * @author Luis José Sánchez
 */
public class Gato extends Animal {

  private String raza;


  public Gato (Sexo s, String r) {
    super(s);
    raza = r;
  }

  public Gato (Sexo s) {
    super(s);
    raza = "siamés";
  }

  public Gato (String r) {
    super(Sexo.HEMBRA);
    raza = r;
  }

  public Gato () {
    super(Sexo.HEMBRA);
    raza = "siamés";
  }

  public String toString() {
    return super.toString()
            + "Raza: " + this.raza
            + "\n*************************\n";
  }

  /**
   * Hace que el gato maulle.
   */
  public void maulla() {
    System.out.println("Miauuuu");
  }

  /**
   * Hace que el gato ronronee
   */
```

```
  public void ronronea() {
    System.out.println("mrrrrrr");
  }

  /**
   * Hace que el gato coma.
   * A los gatos les gusta el pescado, si le damos otra comida
   * la rechazará.
   *
   * @param comida la comida que se le ofrece al gato
   */
  public void come(String comida) {
    if (comida.equals("pescado")) {
      System.out.println("Hmmmm, gracias");
    } else {
      System.out.println("Lo siento, yo solo como pescado");
    }
  }

  /**
   * Pone a pelear dos gatos.
   * Solo se van a pelear dos machos entre sí.
   *
   * @param contrincante es el gato contra el que pelear
   */
  public void peleaCon(Gato contrincante) {
    if (this.getSexo() == Sexo.HEMBRA) {
      System.out.println("no me gusta pelear");
    } else {
      if (contrincante.getSexo() == Sexo.HEMBRA) {
        System.out.println("no peleo contra gatitas");
      } else {
        System.out.println("ven aquí que te vas a enterar");
      }
    }
  }
}
```

Observa que se definen nada menos que cuatro constructores en la clase `Gato`. Desde el programa principal se dilucida cuál de ellos se utiliza en función del número y tipo de parámetros que se pasa al método. Por ejemplo, si desde el programa principal se crea un gato de esta forma

```
Gato gati = new Gato();
```

entonces se llamaría al constructor definido como

```
public Gato () {
  super(Sexo.HEMBRA);
  raza = "siamés";
}
```

Por tanto `gati` sería una gata de raza siamés. Si, por el contrario, creamos `gati` de esta otra manera desde el programa principal

```
Gato gati = new Gato(Sexo.MACHO, "siberiano");
```

se llamaría al siguiente constructor

```
public Gato (Sexo s, String r) {
  super(s);
  raza = r;
}
```

y `gati` sería en este caso un gato macho de raza siberiano.

El método `super()` hace una llamada al método equivalente de la superclase. Fíjate que se utiliza tanto en el constructor como en el método `toString()`. Por ejemplo, al llamar a `super()` dentro del método `toString()` se está llamando al `toString()` que hay definido en la clase `Animal`, justo un nivel por encima de `Gato` en la jerarquía de clases.

A continuación tenemos la definición de la clase `Ave` que es una subclase de `Animal`.

```
/**
 * Ave.java
 * Definición de la clase Ave
 * @author Luis José Sánchez
 */
public class Ave extends Animal {

  public Ave(Sexo s) {
    super(s);
  }

  public Ave() {
    super();
  }

  /**
   * Hace que el ave se limpie.
   */
  public void aseate() {
```

```
    System.out.println("Me estoy limpiando las plumas");
  }

  /**
   * Hace que el ave levante el vuelo.
   */
  public void vuela() {
    System.out.println("Estoy volando");
  }
}
```

Sobrecarga de métodos

Un método se puede **redefinir** (volver a definir con el mismo nombre) en una subclase. Por ejemplo, el método `vuela` que está definido en la clase `Ave` se vuelve a definir en la clase `Pinguino`. En estos casos, indicaremos nuestra intención de sobreescribir un método mediante la etiqueta `@Override`.

Si no escribimos esta etiqueta, la sobreescritura del método se realizará de todas formas ya que `@Override` indica simplemente una intención. Ahora imagina que quieres sobreescribir el método `come` de `Animal` declarando un `come` específico para los gatos en la clase `Gato`. Si escribes `@Override` y luego te equivocas en el nombre del método y escribes `comer`, entonces el compilador diría algo como: “¡Cuidado! algo no está bien, me has dicho que ibas a sobreescribir un método de la superclase y sin embargo `comer` no está definido”.

A continuación tienes la definición de la clase `Pinguino`.

```
/**
 * Pinguino.java
 * Definición de la clase Pinguino
 * @author Luis José Sánchez
 */
public class Pinguino extends Ave {

  public Pinguino() {
    super();
  }

  public Pinguino(Sexo s) {
    super(s);
  }

  /**
   * El pingüino se siente triste porque no puede volar.
```

```
   */
  @Override
  public void vuela() {
    System.out.println("No puedo volar");
  }
}
```

Con el siguiente programa se prueba la clase `Animal` y todas las subclases que derivan de ella. Observa cada línea y comprueba qué hace el programa.

```
/**
 * PruebaAnimal.java
 * Programa que prueba la clase Animal y sus subclases
 * @author Luis José Sánchez
 */
public class PruebaAnimal {
  public static void main(String[] args) {

    Gato garfield = new Gato(Sexo.MACHO, "romano");
    Gato tom = new Gato(Sexo.MACHO);
    Gato lisa = new Gato(Sexo.HEMBRA);
    Gato silvestre = new Gato();

    System.out.println(garfield);
    System.out.println(tom);
    System.out.println(lisa);
    System.out.println(silvestre);

    Ave miLoro = new Ave();
    miLoro.aseate();
    miLoro.vuela();

    Pinguino pingu = new Pinguino(Sexo.HEMBRA);
    pingu.aseate();
    pingu.vuela();
  }
}
```

En el ejemplo anterior, los objetos `miLoro` y `pingu` actúan de manera **polimórfica** porque a ambos se les aplican los métodos `aseate` y `vuela`.



Polimorfismo

En Programación Orientada a Objetos, se llama **polimorfismo** a la capacidad que tienen los objetos de distinto tipo (de distintas clases) de responder al mismo método.

9.6 Atributos y métodos de clase (`static`)

Hasta el momento hemos definido atributos de instancia como `raza`, `sexo` o `color` y métodos de instancia como `maulla`, `come` o `vuela`. De tal modo que si en el programa se crean 20 gatos, cada uno de ellos tiene su propia raza y puede haber potencialmente 20 razas diferentes. También podría aplicar el método `maulla` a todos y cada uno de esos 20 gatos.

No obstante, en determinadas ocasiones, nos puede interesar tener atributos de clase (variables de clase) y métodos de clase. Cuando se define una variable de clase solo existe una copia del atributo para toda la clase y no una para cada objeto. Esto es útil cuando se quiere llevar la cuenta global de algún parámetro. Los métodos de clase se aplican a la clase y no a instancias concretas.

A continuación se muestra un ejemplo que contiene la variable de clase `kilometrajeTotal`. Si bien cada coche tiene un atributo `kilometraje` donde se van acumulando los kilómetros que va recorriendo, en la variable de clase `kilometrajeTotal` se lleva la cuenta de los kilómetros que han recorrido todos los coches que se han creado.

También se crea un método de clase llamado `getKilometrajeTotal` que simplemente es un *getter* para la variable de clase `kilometrajeTotal`.

```java
/**
 * Coche.java
 * Definición de la clase Coche
 * @author Luis José Sánchez
 */
public class Coche {

  // atributo de clase
  private static int kilometrajeTotal = 0;

  // método de clase
  public static int getKilometrajeTotal() {
    return kilometrajeTotal;
  }

  private String marca;
  private String modelo;
  private int kilometraje;

  public Coche(String ma, String mo) {
    marca = ma;
    modelo = mo;
    kilometraje = 0;
  }
```

```
  public int getKilometraje() {
    return kilometraje;
  }

  /**
   * Recorre una determinada distancia.
   *
   * @param km distancia a recorrer en kilómetros
   */
  public void recorre(int km) {
    kilometraje += km;
    kilometrajeTotal += km;
  }
}
```

Como ya hemos comentado, el atributo `kilometrajeTotal` almacena el número total de kilómetros que recorren todos los objetos de la clase `Coche`, es un único valor, por eso se declara como `static`. Por el contrario, el atributo `kilometraje` almacena los kilómetros recorridos por un objeto concreto y tendrá un valor distinto para cada uno de ellos. Si en el programa principal se crean 20 objetos de la clase `Coche`, cada uno tendrá su propio `kilometraje`.

A continuación se muestra el programa que prueba la clase `Coche`.

```
/**
 * PruebaCoche.java
 * Programa que prueba la clase Coche
 * @author Luis José Sánchez
 */
public class PruebaCoche {
  public static void main(String[] args) {

    Coche cocheDeLuis = new Coche("Saab", "93");
    Coche cocheDeJuan = new Coche("Toyota", "Avensis");

    cocheDeLuis.recorre(30);
    cocheDeLuis.recorre(40);
    cocheDeLuis.recorre(220);
    cocheDeJuan.recorre(60);
    cocheDeJuan.recorre(150);
    cocheDeJuan.recorre(90);
    System.out.println("El coche de Luis ha recorrido " + cocheDeLuis.getKilometraje() + "Km");
    System.out.println("El coche de Juan ha recorrido " + cocheDeJuan.getKilometraje() + "Km");
    System.out.println("El kilometraje total ha sido de " + Coche.getKilometrajeTotal() + "Km"\
);
```

```
  }
}
```

El método `getKilometrajeTotal()` se aplica a la clase `Coche` por tratarse de un método de clase (método `static`). Este método no se podría aplicar a una instancia, de la misma manera que un método que no sea `static` no se puede aplicar a la clase sino a los objetos.

9.7 Interfaces

Una **interfaz** contiene únicamente la cabecera de una serie de métodos (opcionalmente también puede contener constantes). Por tanto se encarga de especificar un comportamiento que luego tendrá que ser implementado. La **interfaz** no especifica el "cómo" ya que no contiene el cuerpo de los métodos, solo el "qué".

Una **interfaz** puede ser útil en determinadas circunstancias. En principio, separa la definición de la implementación o, como decíamos antes, el "qué" del "cómo". Tendremos entonces la menos dos ficheros, la **interfaz** y la clase que implementa esa **interfaz**. Se puede dar el caso que un programador escriba la **interfaz** y luego se la pase a otro programador para que sea éste último quien la implemente.

Hay que destacar que cada **interfaz** puede tener varias implementaciones asociadas.

Para ilustrar el uso de interfaces utilizaremos algunas clases ya conocidas. La superclase que va a estar por encima de todas las demás será la clase `Animal` vista con anterioridad. El código de esta clase no varía, por lo tanto no lo vamos a reproducir aquí de nuevo.

Definimos la **interfaz** `Mascota`.

```
/**
 * Mascota.java
 * Definición de la interfaz Mascota
 *
 * @author Luis José Sánchez
 */
public interface Mascota {
  String getCodigo();
  void hazRuido();
  void come(String comida);
  void peleaCon(Animal contrincante);
}
```

Como puedes ver, únicamente se escriben las cabeceras de los métodos que debe tener la/s clase/s que implemente/n la **interfaz** `Mascota`.

Una de las implementaciones de `Mascota` será `Gato`.

```java
/**
 * Gato.java
 * Definición de la clase Gato
 *
 * @author Luis José Sánchez
 */
public class Gato extends Animal implements Mascota {

  private String codigo;

  public Gato (Sexo s, String c) {
    super(s);
    this.codigo = c;
  }

  @Override
  public String getCodigo() {
    return this.codigo;
  }

  /**
   * Hace que el gato emita sonidos.
   */
  @Override
  public void hazRuido() {
    this.maulla();
    this.ronronea();
  }

  /**
   * Hace que el gato maulle.
   */
  public void maulla() {
    System.out.println("Miauuuu");
  }

  /**
   * Hace que el gato ronronee
   */
  public void ronronea() {
    System.out.println("mrrrrrr");
  }

  /**
   * Hace que el gato coma.
   * A los gatos les gusta el pescado, si le damos otra comida
```

```
 * la rechazará.
 *
 * @param comida la comida que se le ofrece al gato
 */
@Override
public void come(String comida) {

  if (comida.equals("pescado")) {
    super.come();
    System.out.println("Hmmmm, gracias");
  } else {
    System.out.println("Lo siento, yo solo como pescado");
  }
}

/**
 * Pone a pelear al gato contra otro animal.
 * Solo se van a pelear dos machos entre sí.
 *
 * @param contrincante es el animal contra el que pelear
 */
@Override
public void peleaCon(Animal contrincante) {
  if (this.getSexo() == Sexo.HEMBRA) {
    System.out.println("no me gusta pelear");
  } else {
    if (contrincante.getSexo() == Sexo.HEMBRA) {
      System.out.println("no peleo contra hembras");
    } else {
      System.out.println("ven aquí que te vas a enterar");
    }
  }
}
}
```

Mediante la siguiente línea:

```
public class Gato extends Animal implements Mascota {
```

estamos diciendo que `Gato` es una subclase de `Animal` y que, además, es una implementación de la **interfaz** `Mascota`. Fíjate que no es lo mismo la herencia que la implementación.

Observa que los métodos que se indicaban en `Mascota` únicamente con la cabecera ahora están implementados completamente en `Gato`. Además, `Gato` contiene otros métodos que no se indicaban en `Mascota` como `maulla` y `ronronea`.

Los métodos de `Gato` que implementan métodos especificados en `Mascota` deben tener la anotación `@Override`.

Como dijimos anteriormente, una **interfaz** puede tener varias implementaciones. A continuación se muestra `Perro`, otra implementación de `Mascota`.

```
/**
 * Perro.java
 * Definición de la clase Perro
 *
 * @author Luis José Sánchez
 */
public class Perro extends Animal implements Mascota {

  private String codigo;

  public Perro (Sexo s, String c) {
    super(s);
    this.codigo = c;
  }

  @Override
  public String getCodigo() {
    return this.codigo;
  }

  /**
   * Hace que el Perro emita sonidos.
   */
  @Override
  public void hazRuido() {
    this.ladra();
  }

  /**
   * Hace que el Perro ladre.
   */
  public void ladra() {
    System.out.println("Guau guau");
  }

  /**
   * Hace que el Perro coma.
   * A los Perros les gusta la carne, si le damos otra comida la rechazará.
   *
   * @param comida la comida que se le ofrece al Perro
   */
```

```
  @Override
  public void come(String comida) {

    if (comida.equals("carne")) {
      super.come();
      System.out.println("Hmmmm, gracias");
    } else {
      System.out.println("Lo siento, yo solo como carne");
    }
  }

  /**
   * Pone a pelear el perro contra otro animal.
   * Solo se van a pelear si los dos son perros.
   *
   * @param contrincante es el animal contra el que pelear
   */
  @Override
  public void peleaCon(Animal contrincante) {
    if (contrincante.getClass().getSimpleName().equals("Perro")) {
      System.out.println("ven aquí que te vas a enterar");
    } else {
      System.out.println("no me gusta pelear");
    }
  }
}
```

Por último mostramos el programa que prueba `Mascota` y sus implementaciones `Gato` y `Perro`.

```
/**
 * PruebaMascota.java
 * Programa que prueba la interfaz Mascota
 *
 * @author Luis José Sánchez
 */
public class PruebaMascota {
  public static void main(String[] args) {

    Mascota garfield = new Gato(Sexo.MACHO, "34569");
    Mascota lisa = new Gato(Sexo.HEMBRA, "96059");
    Mascota kuki = new Perro(Sexo.HEMBRA, "234678");
    Mascota ayo = new Perro(Sexo.MACHO, "778950");

    System.out.println(garfield.getCodigo());
    System.out.println(lisa.getCodigo());
```

```
    System.out.println(kuki.getCodigo());
    System.out.println(ayo.getCodigo());
    garfield.come("pescado");
    lisa.come("hamburguesa");
    kuki.come("pescado");
    lisa.peleaCon((Gato)garfield);
    ayo.peleaCon((Perro)kuki);
  }
}
```

Observa que para crear una mascota que es un gato escribimos lo siguiente:

```
Mascota garfield = new Gato(Sexo.MACHO, "34569");
```

Una **interfaz** no se puede instanciar, por tanto la siguiente línea sería incorrecta:

```
Mascota garfield = new Mascota(Sexo.MACHO, "34569");
```



Interfaces

La interfaz indica “qué” hay que hacer y la implementación especifica “cómo” se hace.

Una interfaz puede tener varias implementaciones.

Una interfaz no se puede instanciar.

La implementación puede contener métodos adicionales cuyas cabeceras no están en su interfaz.

9.8 Arrays de objetos

Del mismo modo que se pueden crear arrays de números enteros, decimales o cadenas de caracteres, también es posible crear arrays de objetos.

Vamos a definir la clase `Alumno`para luego crear un array de objetos de esta clase.

```java
/**
 * Alumno.java
 * Definición de la clase Alumno
 * @author Luis José Sánchez
 */
public class Alumno {
  private String nombre;
  private double notaMedia = 0.0;

  public String getNombre() {
    return nombre;
  }

  public void setNombre(String nombre) {
    this.nombre = nombre;
  }

  public double getNotaMedia() {
    return notaMedia;
  }

  public void setNotaMedia(double notaMedia) {
    this.notaMedia = notaMedia;
  }
}
```

A continuación se define un array de cinco alumnos que posteriormente se rellena. Por último se muestran los datos de los alumnos por pantalla.

Observa que la siguiente línea únicamente define la estructura del array pero no crea los objetos:

```java
Alumno[] alum = new Alumno[5];
```

Cada objeto concreto se crea de forma individual mediante

```java
alum[i] = new Alumno();
```

Aquí tienes el ejemplo completo.

```
/**
 * ArrayDeAlumnosPrincipal.java
 * Programa que prueba un array de la clase Alumno
 * @author Luis José Sánchez
 */
public class ArrayDeAlumnosPrincipal {
  public static void main(String[] args) {

    // Define la estructura, un array de 5 alumnos
    // pero no crea los objetos
    Alumno[] alum = new Alumno[5];

    // Pide los datos de los alumnos /////////////////////////////////

    System.out.println("A continuacion debera introducir el nombre y la nota media de 5 alumno\
s.");

    String nombreIntroducido;
    double notaIntroducida;

    for(int i = 0; i < 5; i++) {

      alum[i] = new Alumno();

      System.out.println("Alumno " + i);

      System.out.print("Nombre: ");
      nombreIntroducido = System.console().readLine();
      (alum[i]).setNombre(nombreIntroducido);

      System.out.print("Nota media: ");
      notaIntroducida = Double.parseDouble(System.console().readLine());
      alum[i].setNotaMedia(notaIntroducida);
    } // for i

    // Muestra los datos de los alumnos /////////////////////////////////

    System.out.println("Los datos introducidos son los siguientes:");

    double  sumaDeMedias = 0;

    for(int i = 0; i < 5; i++) {
      System.out.println("Alumno " + i);
      System.out.println("Nombre: " + alum[i].getNombre());
      System.out.println("Nota media: " + alum[i].getNotaMedia());
      System.out.println("----------------------------");
```

```
      sumaDeMedias += alum[i].getNotaMedia();
    } // for i

    System.out.println("La media global de la clase es " + sumaDeMedias / 5);
  }
}
```

Veamos ahora un ejemplo algo más complejo. Se trata de una gestión típica - alta, baja, listado y modificación - de una colección de discos. Este tipo de programas se suele denominar CRUD (*Create Read Update Delete*).

Primero se define la clase `Disco`.

```
/**
 * Disco.java
 * Definición de la clase Disco
 * @author Luis José Sánchez
 */
public class Disco {
  private String codigo = "LIBRE";
  private String autor;
  private String titulo;
  private String genero;
  private int duracion; // duración total en minutos

  public String getCodigo() {
    return codigo;
  }

  public void setCodigo(String codigo) {
    this.codigo = codigo;
  }

  public String getAutor() {
    return autor;
  }

  public void setAutor(String autor) {
    this.autor = autor;
  }

  public String getGenero() {
    return genero;
  }
```

```
  public void setGenero(String genero) {
    this.genero = genero;
  }

  public String getTitulo() {
    return titulo;
  }

  public void setTitulo(String titulo) {
    this.titulo = titulo;
  }

  public int getDuracion() {
    return duracion;
  }

  public void setDuracion(int duracion) {
    this.duracion = duracion;
  }

  public String toString() {
    String cadena = "\n------------------------------------------";
        cadena += "\nCódigo: " + this.codigo;
        cadena += "\nAutor: " + this.autor;
        cadena += "\nTítulo: " + this.titulo;
        cadena += "\nGénero: " + this.genero;
        cadena += "\nDuración: " + this.duracion;
        cadena += "\n------------------------------------------";

    return cadena;
  }
}
```

A continuación se crea el programa principal. Cada elemento de la colección será un objeto de la clase `Disco`. Se trata, como verás, de una gestión muy simple pero totalmente funcional.

No se incluye en el manual porque ocuparía varias páginas. El código del programa se puede descargar desde el siguiente enlace: https://github.com/LuisJoseSanchez/aprende-java-con-ejercicios/blob/master/ejemplos/09_POO/ColeccionDeDiscos/ColeccionDeDiscosPrincipal.java

9.9 Ejercicios

Conceptos de POO



Ejercicio 1

¿Cuáles serían los atributos de la clase `PilotoDeFormula1`? ¿Se te ocurren algunas instancias de esta clase?



Ejercicio 2

A continuación tienes una lista en la que están mezcladas varias clases con instancias de esas clases. Para ponerlo un poco más difícil, todos los elementos están escritos en minúscula. Di cuáles son las clases, cuáles las instancias, a qué clase pertenece cada una de estas instancias y cuál es la jerarquía entre las clases: paula, goofy, gardfiel, perro, mineral, caballo, tom, silvestre, pirita, rocinante, milu, snoopy, gato, pluto, animal, javier, bucefalo, pegaso, ayudante_de_santa_claus, cuarzo, laika, persona, pato_lucas.



Ejercicio 3

¿Cuáles serían los atributos de la clase `Vivienda`? ¿Qué subclases se te ocurren?



Ejercicio 4

Piensa en la liga de baloncesto, ¿qué 5 clases se te ocurren para representar 5 elementos distintos que intervengan en la liga?



Ejercicio 5

Haz una lista con los atributos que podría tener la clase caballo. A continuación haz una lista con los posibles métodos (acciones asociadas a los caballos).



Ejercicio 6

Lista los atributos de la clase `Alumno` ¿Sería `nombre` uno de los atributos de la clase? Razona tu respuesta.



Ejercicio 7

¿Cuáles serían los atributos de la clase `Ventana` (de ordenador)? ¿cuáles serían los métodos? Piensa en las propiedades y en el comportamiento de una ventana de cualquier programa.

POO en Java



Ejercicio 1

Implementa la clase `Caballo` vista en un ejercicio anterior. Pruébala creando instancias y aplicándole algunos métodos.



Ejercicio 2

Crea la clase `Vehiculo`, así como las clases `Bicicleta` y `Coche` como subclases de la primera. Para la clase `Vehiculo`, crea los atributos de clase `vehiculosCreados`y `kilometrosTotales`, así como el atributo de instancia `kilometrosRecorridos`. Crea también algún método específico para cada una de las subclases. Prueba las clases creadas mediante un programa con un menú como el que se muestra a continuación:

```
VEHÍCULOS
=========
1. Anda con la bicicleta
2. Haz el caballito con la bicicleta
3. Anda con el coche
4. Quema rueda con el coche
5. Ver kilometraje de la bicicleta
6. Ver kilometraje del coche
7. Ver kilometraje total
8. Salir
Elige una opción (1-8):
```



Ejercicio 3

Crea las clases `Animal`, `Mamifero`, `Ave`, `Gato`, `Perro`, `Canario`, `Pinguino` y `Lagarto`. Crea, al menos, tres métodos específicos de cada clase y redefne el/los método/s cuando sea necesario. Prueba las clases creadas en un programa en el que se instancien objetos y se les apliquen métodos.



Ejercicio 4

Crea la clase `Fracción`. Los atributos serán `numerador` y `denominador`. Y algunos de los métodos pueden ser `invierte`, `simplifica`, `multiplica`, `divide`, etc.



Ejercicio 5

Crea la clase Pizza con los atributos y métodos necesarios. Sobre cada pizza se necesita saber el tamaño - mediana o familiar - el tipo - margarita, cuatro quesos o funghi - y su estado - pedida o servida. La clase debe almacenar información sobre el número total de pizzas que se han pedido y que se han servido. Siempre que se crea una pizza nueva, su estado es "pedida". El siguiente código del programa principal debe dar la salida que se muestra:

```
public class PedidosPizza {
  public static void main(String[] args) {
    Pizza p1 = new Pizza("margarita", "mediana");
    Pizza p2 = new Pizza("funghi", "familiar");
    p2.sirve();
    Pizza p3 = new Pizza("cuatro quesos", "mediana");
    System.out.println(p1);
    System.out.println(p2);
    System.out.println(p3);
    p2.sirve();
    System.out.println("pedidas: " + Pizza.getTotalPedidas());
    System.out.println("servidas: " + Pizza.getTotalServidas());
  }
}
```

```
pizza margarita mediana, pedida
pizza funghi familiar, servida
pizza cuatro quesos mediana, pedida
esa pizza ya se ha servido
pedidas: 3
servidas: 1
```



Ejercicio 6

Crea la clase `Tiempo` con los métodos `suma` y `resta`. Los objetos de la clase `Tiempo` son intervalos de tiempo y se crean de la forma `Tiempo t = new Tiempo(1, 20, 30)` donde los parámetros que se le pasan al constructor son las horas, los minutos y los segundos respectivamente. Crea el método `toString` para ver los intervalos de tiempo de la forma `10h 35m 5s`. Si se suman por ejemplo `30m 40s` y `35m 20s` el resultado debería ser `1h 6m 0s`. Realiza un programa de prueba para comprobar que la clase funciona bien.



Ejercicio 7

Queremos gestionar la venta de entradas (no numeradas) de **Expocoches Campanillas** que tiene 3 zonas, la sala principal con 1000 entradas disponibles, la zona de compra-venta con 200 entradas disponibles y la zona vip con 25 entradas disponibles. Hay que controlar que existen entradas antes de venderlas.

La clase Zona con sus atributos y métodos se muestra a continuación:

```
public class Zona {

  private int entradasPorVender;

  public Zona(int n){
    entradasPorVender = n;
  }

  public int getEntradasPorVender() {
    return entradasPorVender;
  }

  /**
   * Vende un número de entradas.
   * Comprueba si quedan entradas libres antes de realizar la venta.
   *
   * @param n número de entradas a vender
   */
  public void vender(int n) {

    if (this.entradasPorVender == 0) {
      System.out.println("Lo siento, las entradas para esa zona están agotadas.");
    } else if (this.entradasPorVender < n) {
      System.out.println("Sólo me quedan " + this.entradasPorVender
                         + " entradas para esa zona.");
    }

    if (this.entradasPorVender >= n) {
      entradasPorVender -= n;
      System.out.println("Aquí tiene sus " + n + " entradas, gracias.");
    }
  }
}
```

El menú del programa debe ser el que se muestra a continuación. Cuando elegimos la opción 2, se nos debe preguntar para qué zona queremos las entradas y cuántas queremos. Lógicamente, el programa debe controlar que no se puedan vender más entradas de la cuenta.

```
1. Mostrar número de entradas libres
2. Vender entradas
3. Salir
```



Ejercicio 8

Implementa la clase `Terminal`. Un terminal tiene asociado un número. Los terminales se pueden llamar unos a otros y el tiempo de conversación corre para ambos. A continuación se proporciona el contenido del `main` y el resultado que debe aparecer por pantalla.

Programa principal:

```
Terminal t1 = new Terminal("678 11 22 33");
Terminal t2 = new Terminal("644 74 44 69");
Terminal t3 = new Terminal("622 32 89 09");
Terminal t4 = new Terminal("664 73 98 18");
System.out.println(t1);
System.out.println(t2);
t1.llama(t2, 320);
t1.llama(t3, 200);
System.out.println(t1);
System.out.println(t2);
System.out.println(t3);
System.out.println(t4);
```

Salida:

```
Nº 678 11 22 33 - 0s de conversación
Nº 644 74 44 69 - 0s de conversación
Nº 678 11 22 33 - 520s de conversación
Nº 644 74 44 69 - 320s de conversación
Nº 622 32 89 09 - 200s de conversación
Nº 664 73 98 18 - 0s de conversación
```



Ejercicio 9

Implementa la clase `Movil` como subclase de `Terminal` (la clase del ejercicio anterior que ya no hace falta modificar). Cada móvil lleva asociada una tarifa que puede ser “rata”, “mono” o “bisonte”. El coste por minuto es de 6, 12 y 30 céntimos respectivamente. Se tarifican los segundos exactos. Obviamente, cuando un móvil llama a otro, se le cobra al que llama, no al que recibe la llamada. A continuación se proporciona el contenido del main y el resultado que debe aparecer por pantalla. Para que el total tarificado aparezca con dos decimales, puedes utilizar `DecimalFormat`.

Programa principal:

```
Movil m1 = new Movil("678 11 22 33", "rata");
Movil m2 = new Movil("644 74 44 69", "mono");
Movil m3 = new Movil("622 32 89 09", "bisonte");
System.out.println(m1);
System.out.println(m2);
m1.llama(m2, 320);
m1.llama(m3, 200);
m2.llama(m3, 550);
System.out.println(m1);
System.out.println(m2);
System.out.println(m3);
```

Salida:

```
Nº 678 11 22 33 - 0s de conversación - tarificados 0,00 euros
Nº 644 74 44 69 - 0s de conversación - tarificados 0,00 euros
Nº 678 11 22 33 - 520s de conversación - tarificados 0,52 euros
Nº 644 74 44 69 - 870s de conversación - tarificados 1,10 euros
Nº 622 32 89 09 - 750s de conversación - tarificados 0,00 euros
```



Ejercicio 10

Las amebas son seres unicelulares de forma cambiante ya que carecen de pared celular. Fagocitan cualquier cosa que se les pone por delante. Crea la clase `Ameba` con el atributo `peso`, un número entero que indica los microgramos que pesa el bicho. Al tratarse de una unidad tan pequeña, no se tienen en cuenta los decimales, será un dato entero. Cuando Dios crea una ameba de la nada – `new Ameba()` – su peso es de 3 microgramos. Al comer, va incrementando su peso; gasta un microgramo en el proceso de fagocitar y el resto hace que aumente de peso. Por ejemplo, si come una partícula de 6 microgramos – por ej. `miAmeba.come(6)` – engordaría 5 microgramos. Una ameba se puede comer a otra ameba. En este caso, sucede lo mismo que anteriormente, se gasta un microgramo en el proceso de fagocitado y el resto lo engorda la ameba que come. Por ejemplo, si una ameba de 7 microgramos se come a una de 4, acaba pesando 10 microgramos. La ameba comida no se destruye sino que se quedaría con un peso de 0 microgramos, una pena de ameba vamos. Posteriormente, una ameba comida podría recuperarse si ella misma come algo. Nótese que el método `come` está sobrecargado.

Programa principal:

```
Ameba a1 = new Ameba();
a1.come(2);
System.out.println(a1);
Ameba a2 = new Ameba();
a2.come(4);
System.out.println(a2);
a1.come(a2);
System.out.println(a1);
System.out.println(a2);
a2.come(3);
System.out.println(a2);
```

Salida:

```
Soy una ameba y peso 4 microgramos.
Soy una ameba y peso 6 microgramos.
Soy una ameba y peso 9 microgramos.
Soy una ameba y peso 0 microgramos.
Soy una ameba y peso 2 microgramos.
```



Ejercicio 11

La empresa **El Corte Islandés** nos ha encargado una aplicación para gestionar las tarjetas regalo. Como primer paso para implementar la aplicación, es necesario crear la clase principal. Implementa la clase **TarjetaRegalo**. Cuando se crea una nueva tarjeta, se le da un saldo y se asigna de forma automática un número de 5 dígitos. Si se intenta gastar más dinero del que tiene la tarjeta, se debe mostrar un mensaje de error. Dos tarjetas regalo se pueden fusionar creando una nueva tarjeta con la suma del saldo que tenga cada una y un nuevo número aleatorio de 5 cifras. Al fusionar dos tarjetas en una, las dos tarjetas originales se quedarían con 0 € de saldo.

Programa principal:

```
TarjetaRegalo t1 = new TarjetaRegalo(100);
TarjetaRegalo t2 = new TarjetaRegalo(120);
System.out.println(t1);
System.out.println(t2);
t1.gasta(45.90);
t2.gasta(5);
t2.gasta(200);
t1.gasta(3.55);
System.out.println(t1);
System.out.println(t2);
TarjetaRegalo t3 = t1.fusionaCon(t2);
System.out.println(t1);
System.out.println(t2);
System.out.println(t3);
```

Salida:

```
Tarjeta nº 67324 - Saldo 100.00€
Tarjeta nº 02788 - Saldo 120.00€
No tiene suficiente saldo para gastar 200.00€
Tarjeta nº 67324 - Saldo 50.55€
Tarjeta nº 02788 - Saldo 115.00€
Tarjeta nº 67324 - Saldo 0.00€
Tarjeta nº 02788 - Saldo 0.00€
Tarjeta nº 59032 - Saldo 165.55€
```



Ejercicio 12

Se quiere informatizar una biblioteca. Crea las clases **Publicacion**, **Libro** y **Revista**. Las clases deben estar implementadas con la jerarquía correcta. Las características comunes de las revistas y de los libros son el código ISBN, el título, y el año de publicación. Los libros tienen además un atributo prestado. Cuando se crean los libros, no están prestados. Las revistas tienen un número. La clase **Libro** debe implementar la interfaz **Prestable** que tiene los métodos **presta**, **devuelve** y **estaPrestado**.

Programa principal:

```
Libro libro1 = new Libro("123456", "La Ruta Prohibida", 2007);
Libro libro2 = new Libro("112233", "Los Otros", 2016);
Libro libro3 = new Libro("456789", "La rosa del mundo", 1995);
Revista revista1 = new Revista("444555", "Año Cero", 2019, 344);
Revista revista2 = new Revista("002244", "National Geographic", 2003, 255);
System.out.println(libro1);
System.out.println(libro2);
System.out.println(libro3);
System.out.println(revista1);
System.out.println(revista2);
libro2.presta();
if (libro2.estaPrestado()) {
  System.out.println("El libro está prestado");
}
libro2.presta();
libro2.devuelve();
if (libro2.estaPrestado()) {
  System.out.println("El libro está prestado");
}
libro3.presta();
System.out.println(libro2);
System.out.println(libro3);
```

Salida:

```
ISBN: 123456, título: La Ruta Prohibida, año de publicación: 2007 (no prestado)
ISBN: 112233, título: Los Otros, año de publicación: 2016 (no prestado)
ISBN: 456789, título: La rosa del mundo, año de publicación: 1995 (no prestado)
ISBN: 444555, título: Año Cero, año de publicación: 2019
ISBN: 002244, título: National Geographic, año de publicación: 2003
El libro está prestado
Lo siento, ese libro ya está prestado.
ISBN: 112233, título: Los Otros, año de publicación: 2016 (no prestado)
ISBN: 456789, título: La rosa del mundo, año de publicación: 1995 (prestado)
```



Ejercicio 13

Implementa la clase **CuentaCorriente**. Cada cuenta corriente tiene un **número de cuenta** de 10 dígitos. Para simplificar, el número de cuenta se genera de forma aleatoria cuando se crea una cuenta nueva. La cuenta se puede crear con un **saldo** inicial; en caso de no especificar **saldo**, se pondrá a cero inicialmente. En una cuenta se pueden hacer **ingresos** y **gastos**. También es posible hacer una **transferencia** entre una cuenta y otra. Se permite el saldo negativo. En el siguiente capítulo se propone un ejercicio como mejora de éste, en el que se pide llevar un registro de los movimientos realizados.

Programa principal:

```
CuentaCorriente cuenta1 = new CuentaCorriente();
CuentaCorriente cuenta2 = new CuentaCorriente(1500);
CuentaCorriente cuenta3 = new CuentaCorriente(6000);
System.out.println(cuenta1);
System.out.println(cuenta2);
System.out.println(cuenta3);
cuenta1.ingreso(2000);
cuenta2.cargo(600);
cuenta3.ingreso(75);
cuenta1.cargo(55);
cuenta2.transferencia(cuenta3, 100);
System.out.println(cuenta1);
System.out.println(cuenta2);
System.out.println(cuenta3);
```

Salida:

```
Número de cta: 6942541557 Saldo: 0,00 €
Número de cta: 9319536518 Saldo: 1500,00 €
Número de cta: 7396941518 Saldo: 6000,00 €
Número de cta: 6942541557 Saldo: 1945,00 €
Número de cta: 9319536518 Saldo: 800,00 €
Número de cta: 7396941518 Saldo: 6175,00 €
```



Ejercicio 14

Implementa la clase **FichaDomino**. Una ficha de dominó tiene dos lados y en cada lado hay un número del 1 al 6 o bien ningún número (blanco). Cuando se crea una ficha, se proporcionan ambos valores. Dos fichas encajan si se pueden colocar una al lado de la otra según el juego del dominó, por ejemplo, las fichas [2 | 5] y [4 | 5] encajan porque se pueden colocar de la forma [2 | 5] [5 | 4]. A continuación se proporciona el contenido del main y el resultado que debe aparecer por pantalla.

Programa principal:

```
FichaDomino f1 = new FichaDomino(6, 1);
FichaDomino f2 = new FichaDomino(0, 4);
FichaDomino f3 = new FichaDomino(3, 3);
FichaDomino f4 = new FichaDomino(0, 1);
System.out.println(f1);
System.out.println(f2);
System.out.println(f3);
System.out.println(f4);
```

```
System.out.println(f2.voltea());
System.out.println(f2.encaja(f4));
System.out.println(f1.encaja(f4));
System.out.println(f1.encaja(f3));
System.out.println(f1.encaja(f2));
```

Salida:

```
[6|1]
[ |4]
[3|3]
[ |1]
[4| ]
true
true
false
false
```



Ejercicio 15

Utiliza la clase anterior para generar una secuencia de 8 fichas creadas de forma aleatoria, que encajen bien y que estén bien colocadas según el juego del dominó. No hay que controlar si se repiten o no las fichas.

Ejemplo:

```
[6|1][1|4][4|4][4| ][ |3][3|2][2|6][6|5]
```



Ejercicio 16

Crea las clases **Punto** y **Linea**. De un punto se tienen que saber sus coordenadas x e y, mientras que una línea está definida por dos puntos. Define las clases y los métodos necesarios para que el siguiente código muestre la salida que se indica.

Programa principal:

```
Punto p1 = new Punto(4.21, 7.3);
Punto p2 = new Punto(-2, 1.66);
Linea l = new Linea(p1, p2);
System.out.println(l);
```

Salida:

```
Línea formada por los puntos (4.21, 7.3) y (-2.0, 1.66)
```



Ejercicio 17

Implementa las clases **Piramide** y **Rectangulo**. Sobre una pirámide se debe saber su **altura** y sobre un rectángulo se debe saber tanto la **base** como la **altura**. Cada una de las clases debe tener un atributo de clase (static) que lleve la cuenta de las pirámides y de los rectángulos creados respectivamente. El siguiente código que va dentro del `main` genera la salida que se indica.

Programa principal:

```
Piramide p = new Piramide(4);
Rectangulo r1 = new Rectangulo(4, 3);
Rectangulo r2 = new Rectangulo(6, 2);
System.out.println(p);
System.out.println(r1);
System.out.println(r2);
System.out.println("Pirámides creadas: " + Piramide.getPiramidesCreadas());
System.out.println("Rectángulos creados: " + Rectangulo.getRectangulosCreados());
```

Salida:

```
   *
  ***
 *****
*******

****
****
****

******
******

Pirámides creadas: 1
Rectángulos creados: 2
```



Ejercicio 18

Una empresa quiere implementar un programa que lleve el control de las incidencias que se producen en sus ordenadores. Cada incidencia tiene un código: 1, 2, 3, 4, etc. Cuando se crea una nueva incidencia, se le asigna un código de forma automática y se pone el estado como **“pendiente”**. Al crear una incidencia hay que indicar también el número de puesto (un número entero). Cuando se resuelve una incidencia, hay que proporcionar información sobre cómo se ha resuelto o qué es lo que fallaba, además, el estado pasa a **“resuelta”**. El siguiente trozo de código que va dentro del `main` genera la salida que se muestra a continuación.

Programa principal:

```
Incidencia inc1 = new Incidencia(105, "No tiene acceso a internet");
Incidencia inc2 = new Incidencia(14, "No arranca");
Incidencia inc3 = new Incidencia(5, "La pantalla se ve rosa");
Incidencia inc4 = new Incidencia(237, "Hace un ruido extraño");
Incidencia inc5 = new Incidencia(111, "Se cuelga al abrir 3 ventanas");
inc2.resuelve("El equipo no estaba enchufado");
inc3.resuelve("Cambio del cable VGA");
System.out.println(inc1);
System.out.println(inc2);
System.out.println(inc3);
System.out.println(inc4);
System.out.println(inc5);
System.out.println("Incidencias pendientes: " + Incidencia.getPendientes());
```

Salida:

```
Incidencia 1 - Puesto: 105 - No tiene acceso a internet - Pendiente
Incidencia 2 - Puesto: 14 - No arranca - Resuelta - El equipo no estaba enchufado
Incidencia 3 - Puesto: 5 - La pantalla se ve rosa - Resuelta - Cambio del cable VGA
Incidencia 4 - Puesto: 237 - Hace un ruido extraño - Pendiente
Incidencia 5 - Puesto: 111 - Se queda colgado al abrir 3 ventanas - Pendiente
Incidencias pendientes: 3
```

Arrays de objetos



Ejercicio 1

Utiliza la clase `Gato` para crear un array de cuatro gatos e introduce los datos de cada uno de ellos mediante un bucle. Muestra a continuación los datos de todos los gatos utilizando también un bucle.



Ejercicio 2

Cambia el programa anterior de tal forma que los datos de los gatos se introduzcan directamente en el código de la forma `gatito[2].setColor("marrón")` o bien mediante el constructor, de la forma `gatito[3] = new Gato("Garfield", "naranja", "macho")`. Muestra a continuación los datos de todos los gatos utilizando un bucle.



Ejercicio 3

Realiza el programa "Colección de discos" por tu cuenta, mirando lo menos posible el ejemplo que se proporciona. Pruébalo primero para ver cómo funciona y luego intenta implementarlo tú mismo.



Ejercicio 4

Modifica el programa "Colección de discos" como se indica a continuación:

a) Mejora la opción "Nuevo disco" de tal forma que cuando se llenen todas las posiciones del array, el programa muestre un mensaje de error. No se permitirá introducir los datos de ningún disco hasta que no se borre alguno de la lista.

b) Mejora la opción "Borrar" de tal forma que se verifique que el código introducido por el usuario existe.

c) Modifica el programa de tal forma que el código del disco sea único, es decir que no se pueda repetir.

d) Crea un submenú dentro dentro de "Listado" de tal forma que exista un listado completo, un listado por autor (todos los discos que ha publicado un determinado autor), un listado por género (todos los discos de un género determinado) y un listado de discos cuya duración esté en un rango determinado por el usuario.



Ejercicio 5

Crea el programa GESTISIMAL (GESTIón SIMplificada de Almacén) para llevar el control de los artículos de un almacén. De cada artículo se debe saber el código, la descripción, el precio de compra, el precio de venta y el stock (número de unidades). El menú del programa debe tener, al menos, las siguientes opciones:

```
1. Listado
2. Alta
3. Baja
4. Modificación
5. Entrada de mercancía
6. Salida de mercancía
7. Salir
```

La entrada y salida de mercancía supone respectivamente el incremento y decremento de stock de un determinado artículo. Hay que controlar que no se pueda sacar más mercancía de la que hay en el almacén.

10. Colecciones y diccionarios

10.1 Colecciones: la clase `ArrayList`

Una colección en Java es una estructura de datos que permite almacenar muchos valores del mismo tipo; por tanto, conceptualmente es prácticamente igual que un *array*. Según el uso y según si se permiten o no repeticiones, Java dispone de un amplio catálogo de colecciones: `ArrayList` (lista), `ArrayBlockingQueue` (cola), `HashSet` (conjunto), `Stack` (pila), etc. En este manual estudiaremos la colección `ArrayList`.

Un `ArrayList` es una estructura en forma de lista que permite almacenar elementos del mismo tipo (pueden ser incluso objetos); su tamaño va cambiando a medida que se añaden o se eliminan esos elementos.

Nos podemos imaginar un `ArrayList` como un conjunto de celdas o cajoncitos donde se guardan los valores, exactamente igual que un *array* convencional. En la práctica será más fácil trabajar con un `ArrayList`.

En capítulos anteriores hemos podido comprobar la utilidad del *array*; es un recurso imprescindible que cualquier programador debe manejar con soltura. No obstante, el *array* presenta algunos inconvenientes. Uno de ellos es la necesidad de conocer el tamaño exacto en el momento de su creación. Una colección, sin embargo, se crea sin que se tenga que especificar el tamaño; posteriormente se van añadiendo y quitando elementos a medida que se necesitan.

Trabajando con *arrays* es frecuente cometer errores al utilizar los índices; por ejemplo al intentar guardar un elemento en una posición que no existe (índice fuera de rango). Aunque las colecciones permiten el uso de índices, no es necesario indicarlos siempre. Por ejemplo, en una colección del tipo `ArrayList`, cuando hay que añadir el elemento `"Amapola"`, se puede hacer simplemente `flores.add("Amapola")`. Al no especificar índice, el elemento `"Amapola"` se añadiría justo al final de `flores` independientemente del tamaño y del número de elementos que se hayan introducido ya.

La clase `ArrayList` es muy similar a la clase `Vector`. Ésta última está obsoleta y, por tanto, no se recomienda su uso.

Principales métodos de `ArrayList`

Las operaciones más comunes que se pueden realizar con un objeto de la clase `ArrayList` son las siguientes:

add(elemento)
    Añade un elemento al final de la lista.

add(indice, elemento)
Inserta un elemento en una posición determinada, desplazando el resto de elementos hacia la derecha.

clear()
Elimina todos los elementos pero no borra la lista.

contains(elemento)
Devuelve `true` si la lista contiene el elemento que se especifica y `false` en caso contrario.

get(indice)
Devuelve el elemento de la posición que se indica entre paréntesis.

indexOf(elemento)
Devuelve la posición de la primera ocurrencia del elemento que se indica entre paréntesis.

isEmpty()
Devuelve `true` si la lista está vacía y `false` en caso de tener algún elemento.

remove(indice)
Elimina el elemento que se encuentra en una posición determinada.

remove(elemento)
Elimina la primera ocurrencia de un elemento.

removeIf(filtro)
Elimina los elementos que cumplen una determinada condición.

set(indice, elemento)
Machaca el elemento que se encuentra en una determinada posición con el elemento que se pasa como parámetro.

size()
Devuelve el tamaño (número de elementos) de la lista.

toArray()
Devuelve un *array* con todos y cada uno de los elementos que contiene la lista.

Puedes consultar todos los métodos disponibles en la documentación oficial de la clase `ArrayList`[1].

[1] http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html

Definición de un `ArrayList` e inserción, borrado y modificación de sus elementos

A continuación se muestra un ejemplo en el que se puede ver cómo se declara un `ArrayList` y cómo se insertan y se extraen elementos.

```
import java.util.ArrayList;

/**
 * Ejemplo de uso de la clase ArrayList
 *
 * @author Luis José Sánchez
 */
public class EjemploArrayList01 {
  public static void main(String[] args) {

    ArrayList<String> a = new ArrayList<String>();

    System.out.println("Nº de elementos: " + a.size());

    a.add("rojo");
    a.add("verde");
    a.add("azul");

    System.out.println("Nº de elementos: " + a.size());

    a.add("blanco");

    System.out.println("Nº de elementos: " + a.size());

    System.out.println("El elemento que hay en la posición 0 es " + a.get(0));
    System.out.println("El elemento que hay en la posición 3 es " + a.get(3));
  }
}
```

Observa que al crear un objeto de la clase `ArrayList` hay que indicar el tipo de dato que se almacenará en las celdas de esa lista. Para ello se utilizan los caracteres < y >. No hay que olvidar los paréntesis del final.



Es necesario importar la clase `ArrayList` para poder crear objetos de esta clase, para ello debe aparecer al principio del programa la línea `import java.util.ArrayList;`. Algunos IDEs (por ej. Netbeans) insertan esta línea de código de forma automática.

En el siguiente ejemplo se muestra un `ArrayList` de números enteros.

```java
import java.util.ArrayList;

/**
 * Ejemplo de uso de la clase ArrayList
 *
 * @author Luis José Sánchez
 */
public class EjemploArrayList011 {
  public static void main(String[] args) {

    ArrayList<Integer> a = new ArrayList<Integer>();

    a.add(18);
    a.add(22);
    a.add(-30);

    System.out.println("Nº de elementos: " + a.size());

    System.out.println("El elemento que hay en la posición 1 es " + a.get(1));
  }
}
```

Se define la estructura de la siguiente manera:

```java
ArrayList<Integer> a = new ArrayList<Integer>();
```

Fíjate que no se utiliza el tipo simple `int` sino el wrapper `Integer`. Recuerda que los wrapper son clases que engloban a los tipos simples y les añaden nuevas funcionalidades (p. ej. permiten tratar a las variables numéricas como objetos). El wrapper de `int` es `Integer`, el de `float` es `Float`, el de `double` es `Double`, el de `long` es `Long`, el de `boolean` es `Boolean` y el de `char` es `Character`.

En el siguiente ejemplo podemos ver cómo extraer todos los elementos de una lista a la manera tradicional, con un bucle `for`.

```java
import java.util.ArrayList;

/**
 * Ejemplo de uso de la clase ArrayList
 *
 * @author Luis José Sánchez
 */
public class EjemploArrayList02 {
  public static void main(String[] args) {

    ArrayList<String> a = new ArrayList<String>();
```

```
    a.add("rojo");
    a.add("verde");
    a.add("azul");
    a.add("blanco");
    a.add("amarillo");

    System.out.println("Contenido de la lista: ");

    for(int i=0; i < a.size(); i++) {
      System.out.println(a.get(i));
    }
  }
}
```

Si estás acostumbrado al `for` clásico, habrás visto que es muy sencillo recorrer todos los elementos del `ArrayList`. No obstante, al trabajar con colecciones es recomendable usar el `for` al estilo `foreach` como se muestra en el siguiente ejemplo. Como puedes ver, la sintaxis es más corta y no se necesita crear un índice para recorrer la estructura.

```
import java.util.ArrayList;

/**
 * Ejemplo de uso de la clase ArrayList
 *
 * @author Luis José Sánchez
 */
public class EjemploArrayList03 {
  public static void main(String[] args) {

    ArrayList<String> a = new ArrayList<String>();

    a.add("rojo");
    a.add("verde");
    a.add("azul");
    a.add("blanco");
    a.add("amarillo");

    System.out.println("Contenido de la lista: ");

    for(String color: a) {
      System.out.println(color);
    }
  }
}
```

Fíjate en estas líneas:

```
for(String color: a) {
  System.out.println(color);
}
```

El significado de este código sería el siguiente:

```
for(String color: a)
```

→ Saca uno a uno todos los elementos de `a` y ve metiéndolos en una variable de nombre `color`.

```
System.out.println(color);
```

→ Muestra por pantalla el contenido de la variable `color`.

Veamos ahora en otro ejemplo cómo eliminar elementos de un `ArrayList`. Se utiliza el método `remove()` y se puede pasar como parámetro el índice del elemento que se quiere eliminar, o bien el valor del elemento. O sea, `a.remove(2)` elimina el elemento que se encuentra en la posición 2 de `a` mientras que `a.remove("blanco")` elimina el valor `"blanco"`.

Es importante destacar que el `ArrayList` se reestructura de forma automática después del borrado de cualquiera de sus elementos.

```
import java.util.ArrayList;

/**
 * Ejemplo de uso de la clase ArrayList
 *
 * @author Luis José Sánchez
 */
public class EjemploArrayList04 {
  public static void main(String[] args) {

    ArrayList<String> a = new ArrayList<String>();

    a.add("rojo");
    a.add("verde");
    a.add("azul");
    a.add("blanco");
    a.add("amarillo");
    a.add("blanco");

    System.out.println("Contenido de la lista: ");
```

```
    for(String color: a) {
      System.out.println(color);
    }

    if (a.contains("blanco")) {
      System.out.println("El blanco está en la lista de colores");
    }

    a.remove("blanco");

    System.out.println("Contenido de la lista después de quitar la " +
                       "primera ocurrencia del color blanco: ");

    for(String color: a) {
      System.out.println(color);
    }

    a.remove(2);
    System.out.println("Contenido de la lista después de quitar el " +
                       "elemento de la posición 2: ");

    for(String color: a) {
      System.out.println(color);
    }
  }
}
```

A continuación se muestra un ejemplo en el que se "machaca" una posición del `ArrayList`. Al hacer `a.set(2, "turquesa")`, se borra lo que hubiera en la posición 2 y se coloca el valor `"turquesa"`. Sería equivalente a hacer `a[2] = "turquesa"` en caso de que `a` fuese un *array* tradicional en lugar de un `ArrayList`.

```
import java.util.ArrayList;

/**
 * Ejemplo de uso de la clase ArrayList
 *
 * @author Luis José Sánchez
 */
public class EjemploArrayList05 {
  public static void main(String[] args) {

    ArrayList<String> a = new ArrayList<String>();

    a.add("rojo");
```

```
    a.add("verde");
    a.add("azul");
    a.add("blanco");
    a.add("amarillo");

    System.out.println("Contenido del vector: ");

    System.out.println(a);

    a.set(2, "turquesa");

    System.out.println("Contenido del vector después de machacar la posición 2: ");

    System.out.println(a);
  }
}
```

El método `add` permite añadir elementos a un `ArrayList` como ya hemos visto. Por ejemplo, `a.add("amarillo")` añade el elemento `"amarillo"` al final de `a`. Este método se puede utilizar también con un índice de la forma `a.add(1, "turquesa")`. En este caso, lo que se hace es insertar en la posición indicada. Lo mejor de todo es que el `ArrayList` se reestructura de forma automática desplazando el resto de elementos.

```
import java.util.ArrayList;

/**
 * Ejemplo de uso de la clase ArrayList
 *
 * @author Luis José Sánchez
 */
public class EjemploArrayList06 {
  public static void main(String[] args) {

    ArrayList<String> a = new ArrayList<String>();

    a.add("rojo");
    a.add("verde");
    a.add("azul");
    a.add("blanco");
    a.add("amarillo");

    System.out.println("Contenido de la lista: ");

    System.out.println(a);

    a.add(1, "turquesa");
```

```
    System.out.println("Contenido de la lista después de insertar en la posición 1: ");

    System.out.println(a);
  }
}
```

Mediante `removeIf` se puede hacer un borrado selectivo según si los elementos de una lista cumplen una determinada condición. Por ejemplo, podemos eliminar todos las palabras que contienen la letra "a".

```
import java.util.ArrayList;

/**
 * Ejemplo de uso de la clase ArrayList.
 * Borrado de elementos con removeIf().
 *
 * @author Luis José Sánchez
 */
public class EjemploArrayList066 {
  public static void main(String[] args) {

    ArrayList<String> a = new ArrayList<String>();

    a.add("rojo");
    a.add("verde");
    a.add("azul");
    a.add("blanco");
    a.add("amarillo");

    System.out.println("Contenido de la lista: ");
    System.out.println(a);

    a.removeIf(palabra -> palabra.contains("a"));

    System.out.println("Contenido de la lista después de borrar las palabras que contienen la \
letra \"a\": ");
    System.out.println(a);

  }
}
```

En el siguiente ejemplo se hace otro borrado selectivo. Se eliminan todos los números menores que 10.

```java
import java.util.ArrayList;

/**
 * Ejemplo de uso de la clase ArrayList.
 * Borrado de elementos con removeIf().
 *
 * @author Luis José Sánchez
 */
public class EjemploArrayList067 {
  public static void main(String[] args) {

    ArrayList<Integer> a = new ArrayList<Integer>();

    a.add(20);
    a.add(7);
    a.add(14);
    a.add(32);
    a.add(3);

    System.out.println("Contenido de la lista: ");
    System.out.println(a);

    a.removeIf(numero -> numero < 10);

    System.out.println("Contenido de la lista después de borrar los menores de 10: ");
    System.out.println(a);

  }
}
```

`ArrayList` de objetos

Una colección `ArrayList` puede contener objetos que son instancias de clases definidas por el programador. Esto es muy útil sobre todo en aplicaciones de gestión para guardar datos de alumnos, productos, libros, etc.

En el siguiente ejemplo, definimos una lista de gatos. En cada celda de la lista se almacenará un objeto de la clase `Gato`.

```
import java.util.ArrayList;

/**
 * Uso de un ArrayList de objetos
 *
 * @author Luis José Sánchez
 */
public class EjemploArrayList07 {
  public static void main(String[] args) {

    ArrayList<Gato> g = new ArrayList<Gato>();

    g.add(new Gato("Garfield", "naranja", "mestizo"));
    g.add(new Gato("Pepe", "gris", "angora"));
    g.add(new Gato("Mauri", "blanco", "manx"));
    g.add(new Gato("Ulises", "marrón", "persa"));

    System.out.println("\nDatos de los gatos:\n");

    for (Gato gatoAux: g) {
      System.out.println(gatoAux+"\n");
    }
  }
}
```

En el siguiente apartado se muestra la definición de la clase `Gato`.

Ordenación de un `ArrayList`

Los elementos de una lista se pueden ordenar con el método `sort`. El formato es el siguiente:

```
Collections.sort(lista);
```

Observa que `sort` es un método de clase que está definido en `Collections`. Para poder utilizar este método es necesario incluir la línea

```
import java.util.Collections;
```

al principio del programa. A continuación se muestra un ejemplo del uso de `sort`.

```
import java.util.ArrayList;
import java.util.Collections;

/**
 * Ordenación de un ArrayList
 *
 * @author Luis José Sánchez
 */
public class EjemploArrayList071 {
  public static void main(String[] args) {

    ArrayList<Integer> a = new ArrayList<Integer>();

    a.add(67);
    a.add(78);
    a.add(10);
    a.add(4);

    System.out.println("\nNúmeros en el orden original:");
    for (int numero: a) {
      System.out.println(numero);
    }

    Collections.sort(a);

    System.out.println("\nNúmeros ordenados:");
    for (int numero: a) {
      System.out.println(numero);
    }
  }
}
```

También es posible ordenar una lista de objetos. En este caso es necesario indicar el criterio de ordenación en la definición de la clase. En el programa principal, se utiliza el método `sort` igual que si se tratase de una lista de números o de palabras como se muestra a continuación.

```
import java.util.ArrayList;
import java.util.Collections;

/**
 * Ordenación de un ArrayList de objetos
 *
 * @author Luis José Sánchez
 */
public class EjemploArrayList08 {
```

```
  public static void main(String[] args) {

    ArrayList<Gato> g = new ArrayList<Gato>();

    g.add(new Gato("Garfield", "naranja", "mestizo"));
    g.add(new Gato("Pepe", "gris", "angora"));
    g.add(new Gato("Mauri", "blanco", "manx"));
    g.add(new Gato("Ulises", "marrón", "persa"));
    g.add(new Gato("Adán", "negro", "angora"));

    Collections.sort(g);

    System.out.println("\nDatos de los gatos ordenados por nombre:");

    for (Gato gatoAux: g) {
      System.out.println(gatoAux+"\n");
    }
  }
}
```

Ahora bien, en la definición de la clase `Gato` hay que indicar de alguna manera cómo se debe realizar la ordenación, ya que Java no sabe de antemano si los gatos se ordenan según el color, el nombre, el peso, etc.

Lo primero que hay que hacer es indicar que los objetos de la clase `Gato` se pueden comparar unos con otros. Para ello, cambiamos la siguiente línea:

```
public class Gato
```

por esta otra:

```
public class Gato implements Comparable<Gato>
```

Lo siguiente y no menos importante es definir el método `compareTo`. Este método debe devolver un 0 si los elementos que se comparan son iguales, un número negativo si el primer elemento que se compara es menor que el segundo y un número positivo en caso contrario. Afortunadamente, las clases `String`, `Integer`, `Double`, etc. ya tienen implementado su propio método `compareTo` así que tenemos hecho lo más difícil. Lo único que deberemos escribir en nuestro código es un `compareTo` con los atributos que queremos comparar.

En el caso que nos ocupa, si queremos ordenar los gatos por nombre, tendremos que implementar el `compareTo` de la clase `Gato` de tal forma que nos devuelva el resultado del `compareTo` de los nombres de los gatos que estamos comparando, de la siguiente manera:

```
public int compareTo(Gato g) {
  return (this.nombre).compareTo(g.getNombre());
}
```

Si en lugar de ordenar por nombre, quisiéramos ordenar por raza, el método `compareTo` de la clase `Gato` sería el siguiente:

```
public int compareTo(Gato g) {
  return (this.raza).compareTo(g.getRaza());
}
```

A continuación se muestra la definición completa de la clase `Gato`.

```
/**
 * Definición de la clase Gato
 *
 * @author
 */
public class Gato implements Comparable<Gato> {
  private String nombre;
  private String color;
  private String raza;

  public Gato(String nombre, String color, String raza) {
    this.nombre = nombre;
    this.color = color;
    this.raza = raza;
  }

  public String getNombre() {
    return nombre;
  }

  public String getRaza() {
    return raza;
  }

  public String toString() {
    return "Nombre: " + this.nombre + "\nColor: " + this.color + "\nRaza: " + this.raza;
  }

  public int compareTo(Gato g) {
    return (this.nombre).compareTo(g.getNombre());
  }
```

```
  public boolean equals(Gato g) {
    return (this.nombre).equals(g.getNombre());
  }
}
```

10.2 Diccionarios: la clase `HashMap`

Imagina un diccionario inglés-español. Queremos saber qué significa la palabra "*stiff*". Sabemos que en el diccionario hay muchas entradas y en cada entrada tenemos una palabra en inglés y su correspondiente traducción al español. Buscando por la "s" encontramos que "*stiff*" significa "agujetas".

Un diccionario en Java funciona exactamente igual. Contiene una serie de elementos que son las entradas que a su vez están formadas por un par (clave, valor). La clave (*key*) permite acceder al valor. No puede haber claves duplicadas. En el ejemplo anterior, la clave sería "*stiff*" y el valor "agujetas".

Java dispone de varios tipos de diccionarios: `HashMap`, `EnumMap`, `Hashtable`, `IdentityHashMap`, `LinkedHashMap`, etc. Nosotros estudiaremos el diccionario `HashMap`.

Principales métodos de `HashMap`

Algunos de los métodos más importantes de la clase `HashMap` son:

get(clave)
: Obtiene el valor correspondiente a una clave. Devuelve `null` si no existe esa clave en el diccionario.

put(clave, valor)
: Añade un par (clave, valor) al diccionario. Si ya había un valor para esa clave, se machaca.

keySet()
: Devuelve un conjunto (set) con todas las claves.

values()
: Devuelve una colección con todos los valores (los valores pueden estar duplicados a diferencia de las claves).

entrySet()
: Devuelve una colección con todos los pares (clave, valor).

containsKey(clave)
: Devuelve `true` si el diccionario contiene la clave indicada y `false` en caso contrario.

getKey()
: Devuelve la clave de la entrada. Se aplica a una sola entrada del diccionario (no al diccionario completo), es decir a una pareja (clave, valor).

Por ejemplo:

```
for (Map.Entry pareja: m.entrySet()) {
  System.out.println(pareja.getKey());
}
```

getValue()
Devuelve el contenido de la entrada. Se aplica a una entrada del diccionario (no al diccionario completo), es decir a una pareja (clave, valor).

Por ejemplo:

```
for (Map.Entry pareja: m.entrySet()) {
  System.out.println(pareja.getValue());
}
```

Definición de un `HashMap` e inserción, borrado y modificación de entradas

Al declarar un diccionario hay que indicar los tipos tanto de la clave como del valor. En el siguiente ejemplo definimos el diccionario `m` que tendrá como clave un número entero y una cadena de caracteres como valor. Este diccionario se declara de esta forma:

```
HashMap<Integer, String> m = new HashMap<Integer, String>();
```

No hay que olvidar importar la clase al principio del programa:

```
import java.util.HashMap;
```

Para insertar una entrada en el diccionario se utiliza el método `put` indicando siempre la clave y el valor. Veamos un ejemplo completo.

```
import java.util.HashMap;

/**
 * Ejemplo de uso de la clase HashMap
 *
 * @author Luis José Sánchez
 */
public class EjemploHashMap01 {
  public static void main(String[] args) {
```

```
    HashMap<Integer, String> m = new HashMap<Integer, String>();

    m.put(924, "Amalia Núñez");
    m.put(921, "Cindy Nero");
    m.put(700, "César Vázquez");
    m.put(219, "Víctor Tilla");
    m.put(537, "Alan Brito");
    m.put(605, "Esteban Quito ");

    System.out.println("Los elementos de m son: \n" + m);
  }
}
```

Para extraer valores se utiliza el método `get`. Se proporciona una clave y el diccionario nos devuelve el valor, igual que un diccionario de verdad. Si no existe ninguna entrada con la clave que se indica, se devuelve `null`.

```
import java.util.HashMap;

/**
 * Ejemplo de uso de la clase HashMap
 *
 * @author Luis José Sánchez
 */
public class EjemploHashMap011 {
  public static void main(String[] args) {

    HashMap<Integer, String> m = new HashMap<Integer, String>();

    m.put(924, "Amalia Núñez");
    m.put(921, "Cindy Nero");
    m.put(700, "César Vázquez");
    m.put(219, "Víctor Tilla");
    m.put(537, "Alan Brito");
    m.put(605, "Esteban Quito ");

    System.out.println(m.get(921));
    System.out.println(m.get(605));
    System.out.println(m.get(888));
  }
}
```

¿Y si queremos extraer todas las entradas? Tenemos varias opciones. Podemos usar el método `print` directamente sobre el diccionario de la forma `System.out.print(diccionario)` como vimos en un ejemplo anterior; de esta manera se muestran por pantalla todas las entradas encerradas entre llaves. También podemos convertir el diccionario en

un `entrySet` (conjunto de entradas) y mostrarlo con `print`; de esta forma se obtiene una salida por pantalla muy parecida a la primera (en lugar de llaves se muestran corchetes). Otra opción es utilizar un `for` para recorrer una a una todas las entradas. En este último caso hay que convertir el diccionario en un `entrySet` ya que no se pueden sacar las entradas directamente del diccionario. Estas dos últimas opciones se ilustran en el siguiente ejemplo.

```
import java.util.HashMap;
import java.util.Map;

/**
 * Ejemplo de uso de la clase HashMap
 *
 * @author Luis José Sánchez
 */
public class EjemploHashMap02 {
  public static void main(String[] args) {

    HashMap<Integer, String> m = new HashMap<Integer, String>();

    m.put(924, "Amalia Núñez");
    m.put(921, "Cindy Nero");
    m.put(700, "César Vázquez");
    m.put(219, "Víctor Tilla");
    m.put(537, "Alan Brito");
    m.put(605, "Esteban Quito ");

    System.out.println("Todas las entradas del diccionario extraídas con entrySet:");
    System.out.println(m.entrySet());

    System.out.println("\nEntradas del diccionario extraídas una a una:");
    for (Map.Entry pareja: m.entrySet()) {
      System.out.println(pareja);
    }
  }
}
```

A continuación se muestra el uso de los métodos `getKey` y `getValue` que extraen la clave y el valor de una entrada respectivamente.

```
import java.util.HashMap;
import java.util.Map;

/**
 * Ejemplo de uso de la clase HashMap
 *
 * @author Luis José Sánchez
 */
public class EjemploHashMap03 {
  public static void main(String[] args) {

    HashMap<Integer, String> m = new HashMap<Integer, String>();

    m.put(924, "Amalia Núñez");
    m.put(921, "Cindy Nero");
    m.put(700, "César Vázquez");
    m.put(219, "Víctor Tilla");
    m.put(537, "Alan Brito");
    m.put(605, "Esteban Quito ");

    System.out.println("Código\tNombre\n------\t------------");

    for (Map.Entry pareja: m.entrySet()) {
      System.out.print(pareja.getKey() + "\t");
      System.out.println(pareja.getValue());
    }
  }
}
```

En el último programa de ejemplo hacemos uso del método `containsKey` que nos servirá para saber si existe o no una determinada clave en un diccionario y del método `get` que, como ya hemos visto, sirve para extraer un valor a partir de su clave.

```
import java.util.HashMap;

/**
 * Ejemplo de uso de la clase HashMap
 *
 * @author Luis José Sánchez
 */
public class EjemploHashMap04 {
  public static void main(String[] args) {

    HashMap<Integer, String> m = new HashMap<Integer, String>();

    m.put(924, "Amalia Núñez");
```

```
    m.put(921, "Cindy Nero");
    m.put(700, "César Vázquez");
    m.put(219, "Víctor Tilla");
    m.put(537, "Alan Brito");
    m.put(605, "Esteban Quito ");

    System.out.print("Por favor, introduzca un código: ");
    int codigoIntroducido = Integer.parseInt(System.console().readLine());

    if (m.containsKey(codigoIntroducido)) {
      System.out.print("El código " + codigoIntroducido + " corresponde a ");
      System.out.println(m.get(codigoIntroducido));
    } else {
      System.out.print("El código introducido no existe.");
    }
  }
}
```

10.3 Ejercicios



Ejercicio 1

Crea un `ArrayList` con los nombres de 6 compañeros de clase. A continuación, muestra esos nombres por pantalla. Utiliza para ello un bucle `for` que recorra todo el `ArrayList` sin usar ningún índice.



Ejercicio 2

Realiza un programa que introduzca valores aleatorios (entre 0 y 100) en un `ArrayList` y que luego calcule la suma, la media, el máximo y el mínimo de esos números. El tamaño de la lista también será aleatorio y podrá oscilar entre 10 y 20 elementos ambos inclusive.



Ejercicio 3

Escribe un programa que ordene 10 números enteros introducidos por teclado y almacenados en un objeto de la clase `ArrayList`.



Ejercicio 4

Realiza un programa equivalente al anterior pero en esta ocasión, el programa debe ordenar palabras en lugar de números.



Ejercicio 5

Realiza de nuevo el ejercicio de la colección de discos pero utilizando esta vez una lista para almacenar la información sobre los discos en lugar de un *array* convencional. Comprobarás que el código se simplifica notablemente ¿Cuánto ocupa el programa original hecho con un *array*? ¿Cuánto ocupa este nuevo programa hecho con una lista?



Ejercicio 6

Implementa el control de acceso al área restringida de un programa. Se debe pedir un nombre de usuario y una contraseña. Si el usuario introduce los datos correctamente, el programa dirá "Ha accedido al área restringida". El usuario tendrá un máximo de 3 oportunidades. Si se agotan las oportunidades el programa dirá "Lo siento, no tiene acceso al área restringida". Los nombres de usuario con sus correspondientes contraseñas deben estar almacenados en una estructura de la clase `HashMap`.



Ejercicio 7

La máquina *Eurocoin* genera una moneda de curso legal cada vez que se pulsa un botón siguiendo la siguiente pauta: o bien coincide el valor con la moneda anteriormente generada - 1 céntimo, 2 céntimos, 5 céntimos, 10 céntimos, 25 céntimos, 50 céntimos, 1 euro o 2 euros - o bien coincide la posición – cara o cruz. Simula, mediante un programa, la generación de 6 monedas aleatorias siguiendo la pauta correcta. Cada moneda generada debe ser una instancia de la clase `Moneda` y la secuencia se debe ir almacenando en una lista.

```
Ejemplo:
2 céntimos - cara
2 céntimos - cruz
50 céntimos - cruz
1 euro - cruz
1 euro - cara
10 céntimos - cara
```



Ejercicio 8

Realiza un programa que escoja al azar 10 cartas de la baraja española (10 objetos de la clase `Carta`). Emplea un objeto de la clase `ArrayList` para almacenarlas y asegúrate de que no se repite ninguna.



Ejercicio 9

Modifica el programa anterior de tal forma que las cartas se muestren ordenadas. Primero se ordenarán por palo: bastos, copas, espadas, oros. Cuando coincida el palo, se ordenará por número: as, 2, 3, 4, 5, 6, 7, sota, caballo, rey.



Ejercicio 10

Crea un mini-diccionario español-inglés que contenga, al menos, 20 palabras (con su correspondiente traducción). Utiliza un objeto de la clase `HashMap` para almacenar las parejas de palabras. El programa pedirá una palabra en español y dará la correspondiente traducción en inglés.



Ejercicio 11

Realiza un programa que escoja al azar 5 palabras en español del mini-diccionario del ejercicio anterior. El programa irá pidiendo que el usuario teclee la traducción al inglés de cada una de las palabras y comprobará si son correctas. Al final, el programa deberá mostrar cuántas respuestas son válidas y cuántas erróneas.



Ejercicio 12

Escribe un programa que genere una secuencia de 5 cartas de la baraja española y que sume los puntos según el juego de la brisca. El valor de las cartas se debe guardar en una estructura `HashMap` que debe contener parejas (figura, valor), por ejemplo ("caballo", 3). La secuencia de cartas debe ser una estructura de la clase `ArrayList` que contiene objetos de la clase `Carta`. El valor de las cartas es el siguiente: as → 11, tres → 10, sota → 2, caballo → 3, rey → 4; el resto de cartas no vale nada.

```
Ejemplo:
as de oros
cinco de bastos
caballo de espadas
sota de copas
tres de oros
Tienes 26 puntos
```



Ejercicio 13

Modifica el programa **Gestisimal** realizado anteriormente añadiendo las siguientes mejoras:

- Utiliza una lista en lugar de un *array* para el almacenamiento de los datos.
- Comprueba la existencia del código en el alta, la baja y la modificación de artículos para evitar errores.
- Cambia la opción "Salida de stock" por "Venta". Esta nueva opción permitirá hacer una venta de varios artículos y emitir la factura correspondiente. Se debe preguntar por los códigos y las cantidades de cada artículo que se quiere comprar. Aplica un 21% de IVA.



Ejercicio 14

Un supermercado de productos ecológicos nos ha pedido hacer un programa para vender su mercancía. En esta primera versión del programa se tendrán en cuenta los productos que se indican en la tabla junto con su precio. Los productos se venden en bote, brick, etc. Cuando se realiza la compra, hay que indicar el producto y el número de unidades que se compran, por ejemplo **"guisantes"** si se quiere comprar un bote de guisantes y la cantidad, por ejemplo **"3"** si se quieren comprar 3 botes. La compra se termina con la palabra **"fin**. Suponemos que el usuario no va a intentar comprar un producto que no existe. Utiliza un diccionario para almacenar los nombres y precios de los productos y una o varias listas para almacenar la compra que realiza el usuario.

A continuación se muestra una tabla con los productos disponibles y sus respectivos precios:

avena	**garbanzos**	**tomate**	**jengibre**	**quinoa**	**guisantes**
2,21	2,39	1,59	3,13	4,50	1,60

Ejemplo:

```
Producto: tomate
Cantidad: 1
Producto: quinoa
Cantidad: 2
Producto: avena
Cantidad: 1
Producto: tomate
Cantidad: 2
Producto: fin

Producto Precio Cantidad Subtotal
---------------------------------
tomate    1,59     1       1,59
quinoa    4,50     2       9,00
avena     2,21     1       2,21
tomate    1,59     2       3,18
---------------------------------
TOTAL: 15,98
```



Ejercicio 15

Realiza una nueva versión del ejercicio anterior con las siguientes mejoras: Si algún producto se repite en diferentes líneas, se deben agrupar en una sola. Por ejemplo, si se pide primero 1 bote de tomate y luego 3 botes de tomate, en el extracto se debe mostrar que se han pedido 4 botes de tomate. Después de teclear "fin", el programa pide un código de descuento. Si el usuario introduce el código "ECODTO", se aplica un 10% de descuento en la compra.

Ejemplo:

```
Producto: tomate
Cantidad: 1
Producto: quinoa
Cantidad: 2
Producto: avena
Cantidad: 1
Producto: quinoa
Cantidad: 2
Producto: tomate
Cantidad: 2
Producto: fin
Introduzca código de descuento (INTRO si no tiene ninguno): ECODTO

Producto Precio Cantidad Subtotal
---------------------------------
tomate    1,59     3       4,77
quinoa    4,50     4      18,00
avena     2,21     1       2,21
---------------------------------
Descuento: 2,50
---------------------------------
TOTAL: 22,48
```



Ejercicio 16

Realiza un programa que sepa decir la capital de un país (en caso de conocer la respuesta) y que, además, sea capaz de aprender nuevas capitales. En principio, el programa solo conoce las capitales de España, Portugal y Francia. Estos datos deberán estar almacenados en un diccionario. Los datos sobre capitales que vaya aprendiendo el programa se deben almacenar en el mismo diccionario. El usuario sale del programa escribiendo la palabra "salir".

Ejemplo:

```
Escribe el nombre de un país y te diré su capital: España
La capital de España es Madrid
Escribe el nombre de un país y te diré su capital: Alemania
No conozco la respuesta ¿cuál es la capital de Alemania?: Berlín
Gracias por enseñarme nuevas capitales
Escribe el nombre de un país y te diré su capital: Portugal
La capital de Portugal es Lisboa
Escribe el nombre de un país y te diré su capital: Alemania
La capital de Alemania es Berlín
Escribe el nombre de un país y te diré su capital: Francia
La capital de Francia es París
Escribe el nombre de un país y te diré su capital: salir
```



Ejercicio 17

Una empresa de venta por internet de productos electrónicos nos ha encargado implementar un carrito de la compra. Crea la clase **Carrito**. Al carrito se le pueden ir agregando elementos que se guardarán en una lista, por tanto, deberás crear la clase **Elemento**. Cada elemento del carrito deberá contener el nombre del **producto**, su **precio** y la **cantidad** (número de unidades de dicho producto). A continuación se muestra tanto el contenido del programa principal como la salida que debe mostrar el programa. Los métodos a implementar se pueden deducir del main.

Contenido del main:

```
Carrito miCarrito = new Carrito();
miCarrito.agrega(new Elemento("Tarjeta SD 64Gb", 19.95, 2));
miCarrito.agrega(new Elemento("Canon EOS 2000D", 449, 1));
System.out.println(miCarrito);
System.out.print("Hay " + miCarrito.numeroDeElementos());
System.out.println(" productos en la cesta.");
System.out.println("El total asciende a "
  + String.format("%.2f", miCarrito.importeTotal()) + " euros");
```

```
System.out.println("\nContinúa la compra...\n");
miCarrito.agrega(new Elemento("Samsung Galaxy Tab", 199, 3));
miCarrito.agrega(new Elemento("Tarjeta SD 64Gb", 19.95, 1));
System.out.println(miCarrito);
System.out.print("Ahora hay " + miCarrito.numeroDeElementos());
System.out.println(" productos en la cesta.");
System.out.println("El total asciende a "
  + String.format("%.2f", miCarrito.importeTotal()) + " euros");
```

Salida:

```
Contenido del carrito
=====================
Tarjeta SD 64Gb PVP: 19,95 Unidades: 2 Subtotal: 39,90
Canon EOS 2000D PVP: 449,00 Unidades: 1 Subtotal: 449,00

Hay 2 productos en la cesta.
El total asciende a 488,90 euros

Continúa la compra...


Contenido del carrito
=====================
Tarjeta SD 64Gb PVP: 19,95 Unidades: 2 Subtotal: 39,90
Canon EOS 2000D PVP: 449,00 Unidades: 1 Subtotal: 449,00
Samsung Galaxy Tab PVP: 199,00 Unidades: 3 Subtotal: 597,00
Tarjeta SD 64Gb PVP: 19,95 Unidades: 1 Subtotal: 19,95

Ahora hay 4 productos en la cesta.
El total asciende a 1105,85 euros
```



Ejercicio 18

Mejora el programa anterior (en otro proyecto diferente) de tal forma que al intentar agregar un elemento al carrito, se compruebe si ya existe el producto y, en tal caso, se incremente el número de unidades sin añadir un nuevo elemento. Observa que en el programa anterior, se repetía el producto "Tarjeta SD 64Gb" dos veces en el carrito. En esta nueva versión ya no sucede esto, si no que se incrementa el número de unidades del producto que se agrega.

El contenido del main es idéntico al ejercicio anterior.

Salida:

```
Contenido del carrito
=====================
Tarjeta SD 64Gb PVP: 19,95 Unidades: 2 Subtotal: 39,90
Canon EOS 2000D PVP: 449,00 Unidades: 1 Subtotal: 449,00

Hay 2 productos en la cesta.
El total asciende a 488,90 euros

Continúa la compra...


Contenido del carrito
=====================
Tarjeta SD 64Gb PVP: 19,95 Unidades: 3 Subtotal: 59,85
Canon EOS 2000D PVP: 449,00 Unidades: 1 Subtotal: 449,00
Samsung Galaxy Tab PVP: 199,00 Unidades: 3 Subtotal: 597,00

Ahora hay 3 productos en la cesta.
El total asciende a 1105,85 euros
```



Ejercicio 19

Realiza un buscador de sinónimos. Utiliza el diccionario español-inglés que se proporciona a continuación. El programa preguntará una palabra y dará una lista de sinónimos (palabras que tienen el mismo significado). Por ejemplo, si se introduce la palabra “caliente”, el programa dará como resultado: ardiente, candente, abrasador. ¿Cómo sabe el programa cuáles son los sinónimos de “caliente”? Muy fácil, en el diccionario debe existir la entrada (“caliente”, “hot”), por tanto solo tendrá que buscar las palabras en español que también signifiquen “hot”; esta información estará en las entradas (“ardiente”, “hot”) y (“abrasador”, “hot”). Cuando una palabra existe en el diccionario pero no tiene sinónimos, debe mostrar el mensaje “No conozco sinónimos de esa palabra”. Si una palabra no está en el diccionario se mostrará el mensaje “No conozco esa palabra”. El usuario sale del programa escribiendo la palabra “salir”.

Español	caliente	rojo	ardiente	verde	agujetas	abrasador	hierro	grande
Inglés	hot	red	hot	green	stiff	hot	iron	big

Ejemplo:

```
Introduzca una palabra y le daré los sinónimos: caliente
Sinónimos de caliente: ardiente, abrasador
Introduzca una palabra y le daré los sinónimos: rojo
No conozco sinónimos de esa palabra
Introduzca una palabra y le daré los sinónimos: blanco
```

```
No conozco esa palabra
Introduzca una palabra y le daré los sinónimos: grande
No conozco sinónimos de esa palabra
Introduzca una palabra y le daré los sinónimos: salir
```



Ejercicio 20

Amplía el programa anterior de tal forma que sea capaz de aprender palabras y sinónimos. Cuando una palabra no tiene sinónimos, es decir, cuando aparece la palabra en español con su traducción y esa traducción no la tiene ninguna otra palabra española, se le preguntará al usuario si quiere añadir uno (un sinónimo) y, en caso afirmativo, se pedirá la palabra y se añadirá al diccionario. Se puede dar la circunstancia de que el usuario introduzca una palabra en español que no está en el diccionario; en tal caso, se mostrará el consiguiente mensaje y se dará la posibilidad al usuario de añadir la entrada correspondiente en el diccionario pidiendo, claro está, la palabra en inglés.

Ejemplo:

```
Introduzca una palabra y le daré los sinónimos: caliente
Sinónimos de caliente: ardiente, abrasador
Introduzca una palabra y le daré los sinónimos: rojo
No conozco sinónimos de esa palabra ¿quiere añadir alguno? (s/n): s
Introduzca un sinónimo de rojo: colorado
Gracias por enseñarme nuevos sinónimos.
Introduzca una palabra y le daré los sinónimos: blanco
No conozco esa palabra ¿quiere añadirla al diccionario? (s/n): s
Introduzca la traducción de blanco en inglés: white
Introduzca una palabra y le daré los sinónimos: rojo
Sinónimos de rojo: colorado
Introduzca una palabra y le daré los sinónimos: blanco
No conozco sinónimos de esa palabra ¿quiere añadir alguno? (s/n): s
Introduzca un sinónimo de blanco: albino
Gracias por enseñarme nuevos sinónimos.
Introduzca una palabra y le daré los sinónimos: blanco
Sinónimos de blanco: albino
Introduzca una palabra y le daré los sinónimos: salir
```



Ejercicio 21

La asociación “Amigos de los anfibios” nos ha encargado una aplicación educativa sobre estos animalitos. Crea un programa que pida al usuario el tipo de anfibio y que, a continuación, nos muestre su hábitat y su alimentación. Si el tipo de anfibio introducido no existe, se debe mostrar el mensaje “Ese tipo de anfibio no existe”.

Ejemplo 1:

```
Introduzca el tipo de anfibio: salamandra
Hábitat: Ecosistemas húmedos.
Alimentación: Pequeños crustáceos e insectos.
```

Ejemplo 2:

```
Introduzca el tipo de anfibio: gato
Ese tipo de anfibio no existe.
```

La información se debe guardar en dos diccionarios (dos HashMap). Uno de ellos tendrá parejas clave-valor del tipo **(tipo de anfibio, hábitat)** y otro **(tipo de anfibio, alimentación)**. A continuación se muestra la información en una tabla:

Tipo de anfibio	rana	salamandra	sapo	tritón
Hábitat	En los trópicos y cerca de las zonas húmedas y acuáticas	Ecosistemas húmedos.	En cualquier sitio salvo el desierto y la Antártida.	América y África.
Alimentación	Larvas e insectos.	Pequeños crustáceos e insectos.	Insectos, lombrices y pequeños roedores.	Insectos



Ejercicio 22

Amplía el ejercicio 13 del capítulo anterior que implementaba cuentas corrientes de un banco de tal forma que cada cuenta lleve un registro de todos los movimientos realizados: ingresos, cargos y transferencias (tanto enviadas como recibidas).

Contenido del `main`

```
CuentaCorriente cuenta1 = new CuentaCorriente();
CuentaCorriente cuenta2 = new CuentaCorriente(1500);
CuentaCorriente cuenta3 = new CuentaCorriente(6000);
cuenta1.ingreso(2000);
cuenta1.cargo(600);
cuenta3.ingreso(75);
cuenta1.cargo(55);
cuenta2.transferencia(cuenta1, 100);
cuenta1.transferencia(cuenta3, 250);
cuenta3.transferencia(cuenta1, 22);
cuenta1.movimientos();
```

Salida

```
Movimientos de la cuenta 1654432813
-----------------------------------
Ingreso de 2000 €  Saldo: 2000,00 €
Cargo de 600 €  Saldo: 1400,00 €
Cargo de 55 €  Saldo: 1345,00 €
Transf. recibida de 100 € de la cuenta 1654432813  Saldo 1445,00 €
Transf. emitida de 250 € a la cuenta 6546817008  Saldo 1195,00 €
Transf. recibida de 22 € de la cuenta 1654432813  Saldo 1217,00 €
```



Ejercicio 23

En ajedrez, el valor de las piezas se mide en peones. Una **dama** vale 9 peones, una **torre** 5 peones, un **alfil** 3, un **caballo** 2 y un **péon** vale, lógicamente, 1 peón. Realiza un programa que genere al azar las capturas que ha hecho un jugador durante una partida. El número de capturas será un valor aleatorio entre 0 y 15. Hay que tener en cuenta que cada jugador tiene la posibilidad de capturar algunas de las siguientes piezas (no más): 1 dama, 2 torres, 2 alfiles, 2 caballos y 8 peones. Al final debe aparecer la puntuación total.

Ejemplo:

```
Fichas capturadas por el jugador:
Alfil (3 peones)
Caballo (2 peones)
Peón (1 peones)
Torre (5 peones)
Peón (1 peones)
Puntos totales: 12 peones.
```

11. Ficheros de texto y paso de parámetros por línea de comandos

Mediante un programa en Java se puede acceder al contenido de un fichero grabado en un dispositivo de almacenamiento (por ejemplo en el disco duro) tanto para leer como para escribir (grabar) datos.

La información contenida en las variables, los arrays, los objetos o cualquier otra estructura de datos es volátil, es decir, se pierde cuando se cierra el programa. Los ficheros permiten tener ciertos datos almacenados y disponibles en cualquier momento para cuando nuestro programa los necesite.

Pensemos por ejemplo en un juego. Podríamos utilizar un fichero para guardar el *ranking* de jugadores con las mejores puntuaciones. De esta manera, estos datos no se perderían al salir del juego. De igual forma, en un programa de gestión, puede ser útil tener un fichero de configuración con datos como el número máximo de elementos que se muestran en un listado o los distintos tipos de IVA aplicables a las facturas. Al modificar esta información, quedará almacenada en el fichero correspondiente y no se perderá cuando se cierre el programa.



Cuando un programa se cierra, se pierde la información almacenada en variables, arrays, objetos o cualquier otra estructura. Si queremos conservar ciertos datos, debemos guardarlos en ficheros.

Hay programas que hacen un uso intensivo de datos. Por ejemplo, la aplicación **Séneca** - que, por cierto, está hecha en Java - lleva el control de la matriculación de cientos de miles de alumnos y la gestión de decenas de miles de profesores. En este caso y en otros similares se utiliza un sistema gestor de bases de datos. El acceso a bases de datos mediante Java lo estudiaremos en el capítulo 13.



La creación y uso de ficheros desde un programa en Java se lleva a cabo cuando hay poca información que almacenar o cuando esa información es heterogénea. En los casos en que la información es abundante y homogénea es preferible usar una base de datos relacional (por ejemplo MySQL) en lugar de ficheros.

En este capítulo estudiaremos, además del acceso a ficheros desde un programa en Java, el paso de parámetros desde la línea de comandos. Son cosas diferentes y no hay que utilizarlas juntas necesariamente pero, como podrás comprobar, se combinan muy bien, por eso hemos incluido estos dos recursos de Java en un mismo capítulo.

11.1 Lectura de un fichero de texto

Aunque Java puede manejar también ficheros binarios, vamos a centrarnos exclusivamente en la utilización de ficheros de texto. Los ficheros de texto tienen una gran ventaja, se pueden crear y manipular mediante cualquier editor como por ejemplo **GEdit**.

Siempre que vayamos a usar ficheros, deberemos incluir al principio del programa una o varias líneas para cargar las clases necesarias. Anque se pueden cargar varias clases en una sola línea usando el asterisco de la siguiente manera:

```
import java.io.*;
```

nosotros no lo haremos así, ya que nuestro código sigue las normas del estándar de Google y estas normas lo prohíben taxativamente. Se importarán las clases usando varias líneas, una línea por cada clase que se importa en el programa; de esta forma:

```
import java.io.BufferedReader;
import java.io.FileReader;
```

No es necesario saber de memoria los nombres de las clases, el entorno de desarrollo **Netbeans** detecta qué clases hacen falta cargar y añade los `import` de forma automática.

Todas las operaciones que se realicen sobre ficheros deberán estar incluidas en un bloque `try-catch`. Esto nos permitirá mostrar mensajes de error y terminar el programa de una forma ordenada en caso de que se produzca algún fallo - el fichero no existe, no tenemos permiso para acceder a él, etc.

El bloque `try-catch` tiene el siguiente formato:

```
try {

  Operaciones_con_fichero

} catch (tipo_de_error  nombre_de_variable) {

  Mensaje_de_error

}
```

En el capítulo 14 sobre control de excepciones se explica de forma detallada el funcionamiento del bloque `try-catch`.

Tanto para leer como para escribir utilizamos lo que en programación se llama un “manejador de fichero”. Es algo así como una variable que hace referencia al fichero con el que queremos trabajar.

En nuestro ejemplo, el manejador de fichero se llama `bf` y se crea de la siguiente manera:

```
BufferedReader bf = new BufferedReader(new FileReader("malaga.txt"));
```

El fichero con el que vamos a trabajar tiene por nombre `malaga.txt` y contiene información extraída de la Wikipedia sobre la bonita ciudad de Málaga[1]. A partir de aquí, siempre que queramos realizar una operación sobre ese archivo, utilizaremos su manejador de fichero asociado, es decir `bf`.

En el ejemplo que se muestra a continuación, el programa lee el contenido del fichero `malaga.txt` y lo muestra tal cual en pantalla, igual que si hiciéramos en una ventana de terminal `cat malaga.txt`.

Observa que se va leyendo el fichero línea a línea mediante `bf.readLine()` hasta que se acaban las líneas. Cuando no quedan más líneas por leer se devuelve el valor `null`.

```
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

/**
 * Ejemplo de uso de la clase File
 * Lectura de un fichero de texto
 *
 * @author Luis José Sánchez
 */
class EjemploFichero01 {
  public static void main(String[] args) {

    try {
      BufferedReader br = new BufferedReader(new FileReader("malaga.txt"));

      String linea = "";

      while (linea != null) {
        System.out.println(linea);
        linea = br.readLine();
      }
```

[1] http://es.wikipedia.org/wiki/M%C3%A1laga

```
      br.close();

    } catch (FileNotFoundException fnfe) { // qué hacer si no se encuentra el fichero

      System.out.println("No se encuentra el fichero malaga.txt");

    } catch (IOException ioe) { // qué hacer si hay un error en la lectura del fichero

      System.out.println("No se puede leer el fichero malaga.txt");

    }
  }
}
```

Es importante "cerrar el fichero" cuando se han realizado todas las operaciones necesarias sobre él. En este ejemplo, esta acción se ha llevado a cabo con la sentencia `bf.close()`.

A continuación se muestra un programa un poco más complejo. Se trata de una aplicación que pide por teclado un nombre de fichero. Previamente en ese fichero (por ejemplo `numeros.txt`) habremos introducido una serie de números, a razón de uno por línea. Se podrían leer también los números si estuvieran separados por comas o espacios aunque sería un poco más complicado (no mucho más). Los números pueden contener decimales ya que se van a leer como `Double`. Cada número que se lee del fichero se va sumando de tal forma que la suma total estará contenida en la variable `suma`; a la par se va llevando la cuenta de los elementos que se van leyendo en la variable `i`. Finalmente, dividiendo la suma total entre el número de elementos obtenemos la media aritmética de los números contenidos en el fichero.

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

/**
 * Ejemplo de uso de la clase File
 * Calcula la media de los números que se encuentran en un fichero de texto
 *
 * @author Luis José Sánchez
 */
class EjemploFichero08 {
  public static void main(String[] args) {

    System.out.print("Introduzca el nombre del archivo donde se encuentran los números: ");
    String nombreFichero = System.console().readLine();

    try {
```

```
      BufferedReader br = new BufferedReader(new FileReader(nombreFichero));

      String linea = "0";
      int i = 0;
      double suma = 0;

      while (linea != null) {
        i++;
        suma += Double.parseDouble(linea);
        linea = br.readLine();
      }
      i--;

      br.close();

      System.out.println("La media es " + suma / (double)i);

    } catch (IOException ioe) {
      System.out.println(ioe.getMessage());
    }
  }
}
```

11.2 Escritura sobre un fichero de texto

La escritura en un fichero de texto es, si cabe, más fácil que la lectura. Solo hay que cambiar `System.out.print("texto")` por `manejador.write("texto")`. Se pueden incluir saltos de línea, tabuladores y espacios igual que al mostrar un mensaje por pantalla.

Es importante ejecutar `close()` después de realizar la escritura; de esta manera nos aseguramos que se graba toda la información en el disco.

Al realizar escrituras en ficheros con Java hay que tener ciertas precauciones. Cuando toca dar este tema en clase es frecuente que a más de un alumno le empiece a ir lento el ordenador, luego se le queda inutilizado y, por último, ni siquiera le arranca ¿qué ha pasado? Pues que ha estado escribiendo datos en ficheros y por alguna razón, su programa se ha metido en un bucle infinito lo que da como resultado cuelgues y ficheros de varios gigabytes de basura. Por eso es muy importante asegurarse bien de que la información que se va a enviar a un fichero es la correcta ¿cómo? muy fácil, enviándola primero a la pantalla.



Primero a pantalla y luego a fichero

Envía primero a la pantalla todo lo que quieras escribir en el fichero. Cuando compruebes que lo que se ve por pantalla es realmente lo que quieres grabar en el fichero, entonces y solo entonces, cambia `System.out.print("texto")` por `manejador.write("texto")`.

A continuación se muestra un programa de ejemplo que crea un fichero de texto y luego esribe en él tres palabras, una por cada línea.

```java
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;

/**
 * Ejemplo de uso de la clase File
 * Escritura en un fichero de texto
 *
 * @author Luis José Sánchez
 */
class EjemploFichero02 {
  public static void main(String[] args) {

  try {
    BufferedWriter bw = new BufferedWriter(new FileWriter("fruta.txt"));

    bw.write("naranja\n");
    bw.write("mango\n");
    bw.write("chirimoya\n");

    bw.close();

    } catch (IOException ioe) {
      System.out.println("No se ha podido escribir en el fichero");
    }
  }
}
```

11.3 Lectura y escritura combinadas

Las operaciones de lectura y escritura sobre ficheros se pueden combinar de tal forma que haya un flujo de lectura y otro de escritura, uno de lectura y dos de escritura, tres de lectura, etc.

En el ejemplo que presentamos a continuación hay dos flujos de lectura y uno de escritura. Observa que se declaran en total tres manejadores de fichero (dos para lectura y uno para escritura). El programa va leyendo, de forma alterna, una línea de cada fichero - una línea de `fichero1.txt` y otra línea de `fichero2.txt` - mientras queden líneas por leer en alguno de los ficheros; y al mismo tiempo va guardando esas líneas en otro fichero con nombre `mezcla.txt`.

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

/**
 * Ejemplo de uso de la clase File
 * Mezcla de dos ficheros
 *
 * @author Luis José Sánchez
 */
class EjemploFichero03 {
  public static void main(String[] args) {

    try {
      BufferedReader br1 = new BufferedReader(new FileReader("fichero1.txt"));
      BufferedReader br2 = new BufferedReader(new FileReader("fichero2.txt"));
      BufferedWriter bw = new BufferedWriter(new FileWriter("mezcla.txt"));

      String linea1 = "";
      String linea2 = "";

      while ( (linea1 != null) || (linea2 != null) ) {
        linea1 = br1.readLine();
        linea2 = br2.readLine();
        if (linea1 != null) {
          bw.write(linea1 + "\n");
        }
        if (linea2 != null) {
          bw.write(linea2 + "\n");
        }
      }

      br1.close();
      br2.close();
      bw.close();

      System.out.println("Archivo mezcla.txt creado satisfactoriamente.");
```

```
    } catch (IOException ioe) {
      System.out.println("Se ha producido un error de lectura/escritura");
      System.err.println(ioe.getMessage());
    }
  }
}
```

11.4 Otras operaciones sobre ficheros

Además de leer desde o escribir en un fichero, hay otras operaciones relacionadas con los archivos que se pueden realizar desde un programa escrito en Java.

Veremos tan solo un par de ejemplos. Si quieres profundizar más, échale un vistazo a la documentación oficial de la clase File[2].

El siguiente ejemplo muestra por pantalla un listado con todos los archivos que contiene un directorio.

```
import java.io.File;

/**
 * Ejemplo de uso de la clase File
 * Listado de los archivos del directorio actual
 *
 * @author Luis José Sánchez
 */
class EjemploFichero04 {
  public static void main(String[] args) {

    File f = new File(".");  // se indica la ruta entre comillas
                // el punto (.) es el directorio actual

    String[] listaArchivos = f.list();
    for(String nombreArchivo : listaArchivos) {
      System.out.println(nombreArchivo);
    }
  }
}
```

El siguiente programa de ejemplo comprueba si un determinado archivo existe o no mediante `exists()` y, en caso de que exista, lo elimina mediante `delete()`. Si intentáramos borrar un archivo que no existe obtendríamos un error.

[2] http://docs.oracle.com/javase/7/docs/api/java/io/File.html

```java
import java.io.File;

/**
 * Ejemplo de uso de la clase File
 * Comprobación de existencia y borrado de un fichero
 *
 * @author Luis José Sánchez
 */
class EjemploFichero05 {
  public static void main(String[] args) {

    System.out.print("Introduzca el nombre del archivo que desea borrar: ");
    String nombreFichero = System.console().readLine();

    File fichero = new File(nombreFichero);

    if (fichero.exists()) {
      fichero.delete();
      System.out.println("El fichero se ha borrado correctamente.");
    } else {
      System.out.println("El fichero " + nombreFichero + " no existe.");
    }
  }
}
```

11.5 Paso de argumentos por línea de comandos

Como dijimos al comienzo de este capítulo, el paso de argumentos por línea de comandos no está directamente relacionado con los ficheros, aunque es muy frecuente combinar estos dos recursos.

Seguro que has usado muchas veces el paso de argumentos por línea de comandos sin saberlo. Por ejemplo, si tecleas el siguiente comando en una ventana de terminal:

```
head -5 /etc/bash.bashrc
```

se muestran por pantalla las 5 primeras líneas del fichero `bash.bashrc` que se encuentra en el directorio `/etc`. En este caso, `head` es el nombre del programa que estamos ejecutando y los valores `-5` y `/etc/bash.bashrc` son los argumentos.

Volvamos al comienzo, al primer capítulo. ¿Recuerdas cómo ejecutaste tu primer programa en Java? ¡Qué tiempos aquéllos! El programa `HolaMundo` se ejecutaba así:

```
java HolaMundo
```

Para probar los ejemplos que te presentamos más adelante, deberás hacer lo mismo y, además, añadir a continuación y separados por espacios los argumentos que quieres que lea el programa. Vamos a verlo en detalle.

El siguiente programa de ejemplo simplemente muestra por pantalla los argumentos introducidos. Compila el programa y prueba a ejecutar lo siguiente:

```
java EjemploArgumentos06 hola que tal 24 1.2 fin
```

con lo que obtendrás el siguiente resultado:

```
Los argumentos introducidos son:
hola
que
tal
24
1.2
fin
```

Los argumentos han pasado al programa en virtud del parámetro que incluimos en la línea del `main`:

```java
public static void main(String[] args) {
```

Fíjate en lo que hay entre paréntesis: `String[] args`. Se trata de un array de cadenas de caracteres (`String`) donde cada uno de los elementos será un argumento que se ha pasado como parámetro. Si has ejecutado el ejemplo tal y como se ha indicado, `args[0]` vale "hola", `args[1]` vale "que", `args[2]` vale "tal", `args[3]` vale "24", `args[4]` vale "1.2" y `args[5]` vale "fin".

```java
/**
 * Paso de argumentos en la línea de comandos
 *
 * @author Luis José Sánchez
 */
class EjemploArgumentos06 {
  public static void main(String[] args) {
    System.out.println("Los argumentos introducidos son: ");
    for (int i = 0; i < args.length; i++) {
      System.out.println(args[i]);
    }
  }
}
```

Hagamos algo un poco más útil con los argumentos. En el siguiente ejemplo, se suman todos los argumentos que se pasan como parámetros y se muestra por pantalla el resultado de la suma.

Después de compilar, deberás ejecutar desde una ventana de terminal lo siguiente (puedes cambiar los números si quieres):

```
java EjemploArgumentos07 10 36 44
```

con lo que obtendrás este resultado:

```
90
```

es decir, la suma de los números que has pasado como parámetros.

```
/**
 * Paso de argumentos en la línea de comandos
 *
 * @author Luis José Sánchez
 */
class EjemploArgumentos07 {
  public static void main(String[] args) {
    int suma = 0;

    for (int i = 0; i < args.length; i++) {
      suma += Integer.parseInt(args[i]);
    }

    System.out.println(suma);
  }
}
```

Observa que el programa convierte en números enteros todos y cada uno de los argumentos mediante el método `Integer.parseInt()` para poder sumarlos.



Los argumentos recogidos por línea de comandos se guardan siempre en un array de `String`. Cuando sea necesario realizar operaciones matemáticas con esos argumentos habría que convertirlos al tipo adecuado mediante `Integer.parseInt()` o `Double.parseDouble()`.

11.6 Combinación de ficheros y paso de argumentos

Llegó el momento de combinar las operaciones con ficheros con el paso de parámetros por línea de comandos. El ejemplo que mostramos a continuación es parecido al que hemos visto en el apartado anterior; calcula la media de una serie de números, pero esta vez esos números se leen desde un fichero y lo que se pasa como parámetro por la línea de comandos es precisamente el nombre del fichero donde están esos números.

Crea un fichero y nómbralo `numeros.txt` e introduce los siguientes números (es importante que estén separados por un salto de línea para que el programa funcione bien):

```
25
100
44
17
6
8
```

A continuación, compila el programa de ejemplo `EjemploFichero09.java` y teclea lo siguiente en la ventana de terminal:

```
java EjemploFichero09 numeros.txt
```

Aparecerá el siguiente resultado:

```
La media es 33.333333333333336
```

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

/**
 * Ejemplo de uso de la clase File
 * Calcula la media de los números que se encuentran en un fichero de texto.
 * El nombre del fichero se pasa como argumento en la línea de comandos.
 *
 * @author Luis José Sánchez
 */
class EjemploFichero09 {
  public static void main(String[] args) {

    if (args.length != 1) {
```

```
      System.out.println("Este programa calcula la media de los números contenidos en un fiche\
ro.");
      System.out.println("Uso del programa: java EjemploFichero09 FICHERO");
      System.exit(-1); // sale del programa
    }

    try {
      BufferedReader bf = new BufferedReader(new FileReader(args[0]));

      String linea = "0";
      int i = 0;
      double suma = 0;

      while (linea != null) {
        i++;
        suma += Double.parseDouble(linea);
        linea = bf.readLine();
      }
      i--;

      bf.close();

      System.out.println("La media es " + suma/(double)i);

    } catch (IOException e) {
      System.out.println(e.getMessage());
    }
  }
}
```

Observa que el programa comprueba el número de argumentos que se pasan por la línea de comandos mediante `if (args.length != 1)` y si este número es distinto de 1, muestra este mensaje:

```
Este programa calcula la media de los números contenidos en un fichero.
Uso del programa: java EjemploFichero09 FICHERO
```

y sale del programa. De esta manera el usuario sabe exactamente cómo hay que utilizar el programa y cómo hay que pasarle la información.

11.7 Procesamiento de archivos de texto

La posibilidad de realizar desde Java operaciones con ficheros abre muchas posibilidades a la hora de procesar archivos: cambiar una palabra por otra, eliminar ciertos

caracteres, mover de sitio una línea o una palabra, borrar espacios o tabulaciones al final de las líneas o cualquier otra cosa que se nos pueda ocurrir.

Cuando se procesa un archivo de texto, los pasos a seguir son los siguientes:

1. Leer una línea del fichero origen mientras quedan líneas por leer.
2. Modificar la línea (normalmente utilizando los métodos que ofrece la clase `String`).
3. Grabar la línea modificada en el fichero destino.
4. Volver al paso 1.

A continuación tienes algunos métodos de la clase `String` que pueden resultar muy útiles para procesar archivos de texto:

- **charAt(int n)** Devuelve el carácter que está en la posición n-ésima de la cadena. Recuerda que la primera posición es la número 0.
- **indexOf(String palabra)** Devuelve un número que indica la posición en la que comienza una palabra determinada.
- **length()** Devuelve la longitud de la cadena.
- **replace(char c1, char c2)** Devuelve una cadena en la que se han cambiado todas las ocurrencias del carácter `c1` por el carácter `c2`.
- **substring(int inicio,int fin)** Devuelve una subcadena.
- **toLowerCase()** Convierte todas las letras en minúsculas.
- **toUpperCase()** Convierte todas las letras en mayúsculas.

Puedes consultar todos los métodos de la clase `String` en la documentación oficial[3].

A continuación tienes un ejemplo en el que se usan los métodos descritos anteriormente.

```
/**
 * Ejemplos de uso de String
 *
 * @author Luis José Sánchez
 */
public class EjemplosString {
  public static void main(String[] args) {

    System.out.println("\nEjemplo 1");
    System.out.println("En la posición 2 de \"berengena\" está la letra "
                      + "berengena".charAt(2));

    System.out.println("\nEjemplo 2");
    String frase = "Hola caracola.";
```

[3] http://docs.oracle.com/javase/7/docs/api/java/lang/String.html

```
    char[] trozo = new char[10];
    trozo[0] = 'z'; trozo[1] = 'z'; trozo[2] = 'z';
    frase.getChars(2, 7, trozo, 1);
    System.out.print("El array de caracteres vale ");
    System.out.println(trozo);

    System.out.println("\nEjemplo 3");
    System.out.println("La secuencia \"co\" aparece en la frase en la posición "
                    + frase.indexOf("co"));

    System.out.println("\nEjemplo 4");
    System.out.println("La palabra \"murciélago\" tiene "
                    + "murciélago".length() + " letras");

    System.out.println("\nEjemplo 5");
    String frase2 = frase.replace('o', 'u');
    System.out.println(frase2);

    System.out.println("\nEjemplo 6");
    frase2 = frase.substring(3, 10);
    System.out.println(frase2);

    System.out.println("\nEjemplo 7");
    System.out.println(frase.toLowerCase());

    System.out.println("\nEjemplo 8");
    System.out.println(frase.toUpperCase());
  }
}
```

A continuación se muestra un programa que procesa archivos de texto. Lo que hace es cambiar cada tabulador por dos espacios en blanco. Lo hice a propósito cuando estaba escribiendo este manual ya que el estándar de Google para la codificación en Java[4] especifica que la sangría debe ser precisamente de dos espacios. Tenía muchos programas escritos en Java (ejemplos y soluciones a ejercicios) que no cumplían este requisito porque la sangría estaba aplicada con el carácter de tabulación; tenía dos posibilidades, abrir todos y cada uno de los ficheros y cambiarlo a mano, o escribir un programa que lo hiciera. Opté por la segunda opción y aquí tienes el programa ¿verdad que es muy sencillo?

[4] http://google-styleguide.googlecode.com/svn/trunk/javaguide.html

```java
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

/**
 * Cambia los tabuladores por 2 espacios
 * @author Luis José Sánchez
 */
public class EjemploProcesamiento10 {
  public static void main(String[] args) {

    for (int i = 0; i < args.length; i++) {

      System.out.print("Procesando el archivo " + args[i] + "...");

      try {
        // renombra el fichero añadiendo ".tmp"
        File ficheroOriginal = new File(args[i]);
        File ficheroTemporal = new File(args[i] + ".tmp");
        ficheroOriginal.renameTo(ficheroTemporal);

        // lee los datos del archivo temporal
        BufferedReader bf = new BufferedReader(new FileReader(args[i] + ".tmp"));

        // crea un fichero nuevo con el nombre original
        BufferedWriter bw = new BufferedWriter(new FileWriter(args[i]));

        String linea = "";

        while ( linea != null ) {
          linea = bf.readLine();

          if (linea != null) {
            // cambia el tabulador por 2 espacios
            linea = linea.replace("\t", "  ");

            bw.write(linea + "\n");
          }
        }

        bf.close();
        bw.close();
```

```
        // borra el fichero temporal
        ficheroTemporal.delete();

      } catch (IOException ioe) {
        System.out.println("Se ha producido un error de lectura/escritura");
        System.err.println(ioe.getMessage());
      }

      System.out.println("hecho");
    }
  }
}
```

11.8 Ejercicios



Ejercicio 1

Escribe un programa que guarde en un fichero con nombre `primos.dat` los números primos que hay entre 1 y 500.



Ejercicio 2

Realiza un programa que lea el fichero creado en el ejercicio anterior y que muestre los números por pantalla.



Ejercicio 3

Escribe un programa que guarde en un fichero el contenido de otros dos ficheros, de tal forma que en el fichero resultante aparezcan las líneas de los primeros dos ficheros mezcladas, es decir, la primera línea será del primer fichero, la segunda será del segundo fichero, la tercera será la siguiente del primer fichero, etc.

Los nombres de los dos ficheros origen y el nombre del fichero destino se deben pasar como argumentos en la línea de comandos.

Hay que tener en cuenta que los ficheros de donde se van cogiendo las líneas pueden tener tamaños diferentes.



Ejercicio 4

Realiza un programa que sea capaz de ordenar alfabéticamente las palabras contenidas en un fichero de texto. El nombre del fichero que contiene las palabras se debe pasar como argumento en la línea de comandos. El nombre del fichero resultado debe ser el mismo que el original añadiendo la coletilla `sort`, por ejemplo `palabras_sort.txt`. Suponemos que cada palabra ocupa una línea.



Ejercicio 5

Escribe un programa capaz de quitar los comentarios de un programa de Java. Se utilizaría de la siguiente manera:

```
quita_commentarios PROGRAMA_ORIGINAL PROGRAMA_LIMPIO
```

Por ejemplo:

```
quita_comentarios hola.java holav2.java
```

crea un fichero con nombre `holav2.java` que contiene el código de `hola.java` pero sin los comentarios.



Ejercicio 6

Realiza un programa que diga cuántas ocurrencias de una palabra hay en un fichero. Tanto el nombre del fichero como la palabra se deben pasar como argumentos en la línea de comandos.

12. Aplicaciones web en Java (JSP)

Hasta el momento, los programas realizados funcionan en una austera ventana de terminal. No se trata de algo casual, hemos elegido esta manera de trabajar de forma deliberada, para que el estudiante se centre en resolver el problema y no se distraiga dándole sombra a un botón o escogiendo la mejor imagen de fondo para un listado.

Llegamos, no obstante, a un punto en que algunos programas piden a gritos una apariencia más vistosa.

En este capítulo vamos a aprender cómo utilizar páginas web como interfaz para programas en Java. Usaremos JSP (JavaServer Pages) que nos permitirá mezclar código en Java con código HTML. El código en Java que utilizaremos será muy parecido al que hemos venido utilizando hasta ahora. Cambiará únicamente lo relativo a mostrar información por pantalla (ahora se volcará todo a HTML) y la manera en que se introducen los datos, que se realizará mediante formularios.



El lector deberá tener unos conocimientos básicos de HTML para entender bien y sacar provecho de este capítulo. Una web excelente para aprender HTML es W3Schools[1]

El estándar de programación de Google para el código fuente escrito en Java[2] no especifica ninguna regla en cuanto a la manera de nombrar los ficheros correspondientes a las aplicaciones JSP. Por tanto vamos a adoptar las convenciones de Oracle[3] en esta materia. Según estas convenciones, los nombres de los ficheros JSP deben estar escritos en *lowerCamelCase*, es decir, la primera letra siempre es minúscula y, en caso de utilizar varias palabras dentro del nombre, éstas van seguidas una de otra empezando cada una por mayúscula. Finalmente se añade la extensión `.jsp`. Por ejemplo, `listadoSocios.jsp` es un nombre correcto, mientras que `ListadoSocios.jsp`, `listado_socios.jsp` o `Listadosocios.jsp` son incorrectos.

12.1 Hola Mundo en JSP

Para desarrollar aplicaciones en JSP utilizaremos el entorno de desarrollo **Netbeans**. Este IDE se puede descargar de forma gratuita en https://netbeans.org/downloads/ [4]

Sigue estos pasos para crear una aplicación en JSP con **Netbeans**:

[1] http://w3schools.com
[2] http://google-styleguide.googlecode.com/svn/trunk/javaguide.html
[3] http://www.oracle.com/technetwork/articles/javase/code-convention-138726.html
[4] Para más detalles sobre la instalación y configuración de **Netbeans**, ver el Apéndice B.

- En el menú principal, selecciona **Archivo** y a continuación **Nuevo Proyecto**. Se puede hacer más rápido con la combinación **Control + Mayúscula + N**.
- Aparecerá una ventana emergente para elegir el tipo de proyecto. Selecciona **Java Web** en el apartado “Categorías” y **Aplicación Web** en el apartado “Proyectos”. Haz clic en **Siguiente**.
- Ahora hay que darle el nombre al proyecto, lo puedes llamar por ejemplo **Saludo1**. Recuerda no utilizar caracteres especiales ni signos de puntuación. Se creará una carpeta con el mismo nombre que el proyecto que contendrá todos los ficheros necesarios. Haz clic en **Siguiente**.
- En la siguiente ventana tendrás que elegir el servidor, la versión de Java y la ruta. Elige **Glass Fish Server**. En caso de no haber ningún servidor disponible, **Netbeans** permite añadir uno sobre la marcha (se lo descarga y lo instala). Haz clic en **Finalizar**.

A la izquierda, en la ventana de proyectos, debe aparecer el que acabas de crear: **Saludo1**. Navegando por la estructura del proyecto, verás que hay una carpeta con nombre **Web Pages**; en esta carpeta se deberán guardar los archivos correspondientes a la aplicación: archivos jsp, html, css, imágenes, archivos de audio, etc. Para que los archivos estén ordenados, se pueden organizar, a su vez, en subcarpetas dentro de **Web Pages**.

Por defecto, cuando se crea un proyecto JSP, se crea el archivo `index.html`. Elimina este archivo de tu proyecto y luego crea `index.jsp` haciendo clic con el botón derecho en **Web Pages → Nuevo → JSP**. Al indicar el nombre no hay que poner la extensión.

Ya tienes un “Hola Mundo” en inglés. Si lo quieres en español solo tienes que cambiar “Hello World” por “Hola Mundo”, con lo que la aplicación quedaría como sigue:

```
<%-- saludo1.jsp --%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  </head>
  <body>
    <h1>¡Hola Mundo!</h1>
    En esta página se usa únicamente HTML.
  </body>
</html>
```

Para ejecutar la aplicación hay que hacer clic en el botón verde **Ejecutar Proyecto** o pulsar **F6**. En ese momento, **Netbeans** buscará un fichero con el nombre `index.jsp` y lo lanzará en el navegador.

Observa que el programa del ejemplo se llama `saludo1.jsp` y no `index.jsp`. En este caso, para ejecutar la aplicación hay que hacer clic con el botón derecho encima de `saludo1.jsp` y seleccionar `Ejecutar`.

De momento, solo hemos visto código HTML ¿dónde está Java? Bien, vamos a verlo, pero antes hay entender bien cómo funciona JSP.

Una aplicación JSP consistirá en una o varias páginas web que normalmente contendrá código HTML y que llevarán "insertado" código en Java. Este código en Java puede colocarse en cualquier parte del archivo y se delimita mediante las etiquetas `<%` y `%>`. Cuando se pulsa **F6** para ejecutar la aplicación sucede lo siguiente:

1. Todo el código (tanto HTML como JAVA) se envía al servidor (Glass Fish o Apache Tomcat por ejemplo).
2. El servidor deja intacto el código HTML y compila y ejecuta el código en Java. Generalmente el código Java genera código HTML mediante las instrucciones `out.print` y `out.println`.
3. Por último, todo el código (ya solo queda HTML) se envía al navegador que es el que muestra la página. Recuerda que el navegador no entiende Java (entiende Javascript que es otro lenguaje diferente) y es imprescindible que todo el código Java haya sido traducido previamente.

12.2 Mezclando Java con HTML

A continuación se muestra un ejemplo que sí tiene Java. Como ves, el código Java está delimitado por los caracteres `<%` y `%>`.

```
<%-- saludo2.jsp --%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  </head>
  <body>
    <% out.println("<h1>¡Hola Mundo!</h1>"); %>
  </body>
</html>
```

Mediante Java, se puede generar cualquier contenido HTML, por ejemplo para modificar estilos, crear tablas, mostrar imágenes, etc. Trabajando en consola, lo que se incluye en un `print` es exactamente lo que se muestra por pantalla. Ahora, con JSP, lo que se incluye dentro del `print` es "renderizado" a posteriori por el navegador. En el siguiente ejemplo se puede ver claramente este comportamiento.

```
<%-- saludo3.jsp --%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  </head>
  <body>
    <h1>¡Hola Caracola!</h1>
    <% out.print("<b><i>"); %>
    Esta línea se ha puesto en negrita y cursiva mediante Java.
    <% out.print("</i></b>"); %>
  </body>
</html>
```

Es muy útil ver el código fuente que se ha generado una vez que se puede visualizar la aplicación en el navegador. Para ver el código fuente tanto en **Firefox** como en **Chrome**, hay que hacer clic con el botón derecho en la página y seleccionar la opción “Ver código fuente de la página”.

Cuando el contenido que se quiere volcar en HTML es el resultado de una expresión, se pueden utilizar de forma opcional las etiquetas `<%=` y `%>`.

```
<%-- saludo4.jsp --%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  </head>
  <body>
    <h1>¡Hola Caracola!</h1>
    <%= "<b><i>" %>
    Esta línea se ha puesto en negrita y cursiva mediante Java.
    <%= "</i></b>" %>
  </body>
</html>
```

En el código Java que se inserta en las aplicaciones JSP se pueden utilizar todos los elemento del lenguaje vistos anteriormente: bucles, sentencias de control, arrays, diccionarios, etc.

En el siguiente ejemplo se utiliza un bucle `for` para mostrar texto en diferentes tamaños, utilizando las etiquetas desde la `<h6>` (cabecera pequeña) hasta la `<h1>` (cabecera más grande).

```
<%-- pruebaVariable1 --%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  </head>
  <body>
    <%
      for (int i = 1; i < 7; i++)
        out.println("<h" + (7-i) + ">"+ i + "</h" + (7-i) + ">");
    %>
  </body>
</html>
```

El código en Java se puede insertar en cualquier parte dentro del código HTML, e incluso, se pueden insertar varios trozos como se puede ver en el siguiente ejemplo. Aunque una variable se haya definido en una determinada zona, su ámbito de actuación no se ve reducido a ese fragmento de código sino que se puede utilizar posteriormente en otro lugar de la aplicación.

Observa que en el siguiente ejemplo, la definición e inicialización de la variable `x` se realiza en un bloque de código en Java y que luego esta variable se utiliza en otros dos bloques diferentes.

```
<%-- pruebaVariable2 --%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  </head>
  <body>
    <% int x = 3;  %>
    <h<% out.print(x); %>>hola</h<% out.print(x); %>>
  </body>
</html>
```

En ocasiones puede ser útil recabar información del sistema: versión de Java en uso, sistema operativo, nombre de usuario, etc. Estos datos se pueden obtener mediante `System.getProperty(propiedad)`.

```
<%-- muestraInfo.jsp --%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  </head>
  <body>
    <h1>Información del entorno de trabajo</h1>
    <%
      out.print("Fabricante de Java: " + System.getProperty("java.vendor"));
      out.print("<br>Url del fabricante: " + System.getProperty("java.vendor.url"));
      out.print("<br>Versión: " + System.getProperty("java.version"));
      out.print("<br>Sistema operativo: " + System.getProperty("os.name"));
      out.print("<br>Usuario: " + System.getProperty("user.name"));
    %>
  </body>
</html>
```

Para mostrar información en una página de forma ordenada es muy frecuente utilizar tablas. Las tablas son especialmente útiles para mostrar listados obtenidos de una base de datos como veremos en el siguiente capítulo.

En el siguiente ejemplo se muestra una tabla con dos columnas; en la primera columna se muestran los números del 0 al 9 y en la segunda, sus correspondientes cuadrados.

```
<%-- tabla.jsp --%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <body>
    <h1>Ejemplo de tabla</h1>
    <table border="2">
      <tr>
        <td>Número</td><td>Cuadrado</td>
      </tr>
      <%
        for(int i = 0; i < 10; i++) {
          out.println("<tr>");
          out.println("<td>" + i + "</td>");
          out.println("<td>");
          out.println(i * i);
          out.println("</td></tr>");
        }
```

```
      %>
    </table>
  </body>
</html>
```

12.3 Recogida de datos en JSP

En cualquier aplicación web, la introducción de datos se realiza mediante formularios. Aunque se puede hacer todo en la misma página, de momento vamos a tener dos. La primera página contendrá un formulario y será la encargada de recoger la información y enviarla a una segunda página. Esta última página recibirá los datos, realizará una serie de cálculos u operaciones si procede y, por último, mostrará un resultado.

En el primer ejemplo - un proyecto que llamaremos `PasoDeCadena` - tenemos una página con nombre `index.jsp` que contiene un formulario HTML. Este formulario contiene una entrada de texto donde el usuario introducirá una cadena de caracteres. Es muy importante darle un valor a la etiqueta `name`. En el caso que nos ocupa tenemos `name="cadenaIntro"`, por tanto el dato que recoge el formulario se llama de esa manera, sería el equivalente al nombre de la variable en Java. No menos importante es indicar la página que recibirá los datos que recoge el formulario. Esta información se indica con la etiqueta `action`, en nuestro ejemplo `action="frase.jsp"` indica que será el fichero `frase.jsp` el que recibirá `cadenaIntro`.

El fichero `index.jsp` únicamente contiene código HTML, por tanto se podría llamar `index.html` y la aplicación funcionaría exactamente igual.

```
<%-- index.jsp (proyecto PasoDeCadena) --%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Paso de cadena</title>
  </head>
  <body>
    <h1>Pasando una cadena de caracteres</h1>
    <form method="post" action="frase.jsp">
      Introduzca el nombre de una fruta:
      <input type="text" name="cadenaIntro">
      <input type="submit" value="OK">
    </form>
  </body>
</html>
```

Para recoger los datos enviados por un formulario se utiliza `request.getParameter("nombreDeVariable")` donde `nombreDeVariable` es el dato que se envía desde el formulario. En caso de que el dato enviado sea un texto que pueda contener tildes o eñes, hay que especificar la codificación mediante `request.setCharacterEncoding("UTF-8")` antes de recoger el dato.

```
<%-- frase.jsp (proyecto PasoDeCadena) --%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Paso de cadena</title>
  </head>
  <body>
    <% request.setCharacterEncoding("UTF-8"); %>
    Me gusta mucho comer
    <% out.print(request.getParameter("cadenaIntro"));%>
  </body>
</html>
```

En el siguiente ejemplo, llamado `Incrementa5`, se muestra cómo manipular en JSP un dato numérico. La aplicación recoge un número y luego muestra ese número incrementado en 5 unidades.

```
<%-- index.jsp (proyecto Incrementa5) --%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  </head>
  <body>
    <form method="get" action="incrementa5.jsp">
      Introduzca un número (puede tener decimales):
      <input type="text" name="numeroIntro">
      <input type="submit" value="OK">
    </form>
  </body>
</html>
```

Observa en el fichero `incrementa5.jsp` cómo se transforma la cadena de caracteres que se recibe en `numeroIntro` en un dato numérico con el método `Double.parseDouble`; exactamente igual que si estuviéramos leyendo un número desde teclado en la ventana de terminal.

```
<%-- incrementa5.jsp (proyecto Incrementa5) --%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  </head>
  <body>
    El número introducido más cinco es
    <%
      double resultado;
      resultado = Double.parseDouble(request.getParameter("numeroIntro")) + 5;
      out.print(resultado);
    %>
  </body>
</html>
```

El siguiente ejemplo ilustra la recogida y envío de dos variables, `x` e `y`. Observa que ahora el formulario contiene dos entradas de texto, una para cada variable.

```
<%-- index.jsp (proyecto Suma) --%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Suma</title>
  </head>
  <body>
    <h1>Supercalculadora</h1>
    <form method="get" action="resultado.jsp">
      x <input type="text" name="x"/></br>
      y <input type="text" name="y"/></br>
      <input type="submit">
    </form>
  </body>
</html>
```

En el fichero `resultado.jsp` se reciben las variables recogidas en `index.jsp` y se suman. Recuerda que por defecto la información enviada por un formulario es una cadena de caracteres. Si queremos sumar los valores introducidos, debemos transformar las cadenas de caracteres a números enteros con el método `Integer.valueOf()`.

```
<%-- resultado.jsp (proyecto Suma) --%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Suma</title>
  </head>
  <body>
    La suma es
    <%
      int primerNumero = Integer.valueOf(request.getParameter("x"));
      int segundoNumero = Integer.valueOf(request.getParameter("y"));
      out.println(primerNumero + segundoNumero);
    %>
  </body>
</html>
```

El siguiente proyecto de ejemplo es `Animales`. En la página principal (`index.jsp`) se puede seleccionar un animal - gato o caracol - y un número. Una vez introducidos los datos, éstos se envían a `animales.jsp`.

```
<%-- index.jsp (proyecto Animales) --%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Animales</title>
  </head>
  <body>
    <form method="post" action="animales.jsp">
      Seleccione animal a visualizar
      <select name="animal">
        <option>Gato</option>
        <option>Caracol</option>
      </select>
      </br>
      Número de animales <input type="text" name="numero" size="3">
      </br>
      <input type="submit">
    </form>
  </body>
</html>
```

La página `animales.jsp` nos mostrará una imagen del animal elegido repetida el número

de veces que hemos indicado.

```
<%-- animales.jsp (proyecto Animales) --%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Animales</title>
  </head>
  <body>
    <%
      String nombreAnimal = request.getParameter("animal");
      String nombreImagen;
      if (nombreAnimal.equals("Gato")) {
        nombreImagen = "gato.jpg";
      } else {
        nombreImagen = "caracol.jpg";
      }

      int veces = Integer.parseInt(request.getParameter("numero"));

      for (int i = 0; i < veces; i++) {
        out.print("<img src=\"" + nombreImagen +"\" width=\"20%\">");
      }
     %>
  </body>
</html>
```

12.4 POO en JSP

En las aplicaciones realizadas en JSP se pueden incluir clases definidas por el usuario para posteriormente crear objetos de esas clases. Lo habitual es que el fichero que contiene la definición de la clase se encuentre separado del programa principal.

En un nuevo proyecto con nombre **GatosConClase** vamos a crear la clase `Gato`; para ello haz clic derecho sobre el icono del proyecto, selecciona **Nuevo** y luego selecciona **Clase de Java**. El nombre de la clase será `Gato` y el nombre del paquete será por ejemplo `daw1`.

El fichero `Gato.java` quedaría como el que se muestra a continuación.

```java
/* Gato.java (proyecto GatosConClase) */

package daw1;

public class Gato {
  private String nombre;
  private String imagen;

  public Gato(String nombre, String imagen) {
    this.nombre = nombre;
    this.imagen = imagen;
  }

  public String getNombre() {
    return nombre;
  }

  public String getImagen() {
    return imagen;
  }

  @Override
  public String toString() {
    return "<img src='" + imagen + "' width='80'>Hola, soy " + nombre + "<br>";
  }

  public String maulla() {
    return "<img src='" + imagen + "' width='80'>Miauuuuuuuu<br>";
  }

  public String come(String comida) {
    return "<img src='" + imagen + "' width='80'>Estoy comiendo " + comida + "<br>";
  }
}
```

Hemos creado una clase `Gato` muy sencilla que únicamente contiene dos atributos: el nombre del gato y el nombre del fichero de su foto. Como métodos, se han implementado `toString()`, `maulla()` y `come()`.

```
<%-- index.jsp (proyecto GatosConClase) --%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@page import="daw1.Gato"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Gatos con clase</title>
  </head>
  <body>
    <h1>Gatos con clase</h1>
    <hr>
    <%
      Gato g1 = new Gato("Pepe", "gato.jpg");
      Gato g2 = new Gato("Garfield", "garfield.jpg");
      Gato g3 = new Gato("Tom", "tom.png");

      out.println(g1);
      out.println(g2);
      out.println(g3);
      out.println(g1.maulla());
      out.println(g2.come("sardinas"));
    %>
  </body>
</html>
```

El código Java del programa principal es casi idéntico al que tendríamos en un programa para consola.

El siguiente ejemplo - GatosConClaseYBocadillos - es una mejora del anterior. Se añaden estilos[5] para mostrar lo que dicen los gatos, igual que en un cómic.

```
/* Gato.java (proyecto GatosConClaseYBocadillos) */

package daw1;

public class Gato {
  private String nombre;
  private String imagen;

  public Gato(String nombre, String imagen) {
    this.nombre = nombre;
    this.imagen = imagen;
  }
```

[5] https://github.com/LuisJoseSanchez/aprende-java-con-ejercicios/blob/master/ejemplos/12_JSP/GatosConClaseYBocadillos/estilos.css

```
  public String getNombre() {
    return nombre;
  }

  public String getImagen() {
    return imagen;
  }

  @Override
  public String toString() {
    return "<div class=\"acciongato\"><img src=\"" + imagen + "\" width=\"80\"><div class=\"ar\
row_box\"> Hola, soy " + nombre + " </div></div>";
  }

  public String maulla() {
    return "<div class=\"acciongato\"><img src=\"" + imagen + "\" width=\"80\"><div class=\"ar\
row_box\"> Miauuuuuuuu </div></div>";
  }

  public String come(String comida) {
    return "<div class=\"acciongato\"><img src=\"" + imagen + "\" width=\"80\"><div class=\"ar\
row_box\"> Estoy comiendo " + comida + " </div></div>";
  }
}
```

El único añadido al programa principal es una línea situada en la cabecera de la página que carga la hoja de estilos `estilos.css`.

```
<%-- index.jsp (proyecto GatosConClaseYBocadillos) --%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@page import="daw1.Gato"%>
<!DOCTYPE html>
<html>
    <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>Gatos con clase y bocadillos</title>
                  <link rel="stylesheet" type="text/css" href="estilos.css" />
    </head>
    <body>
      <h1>Gatos con clase</h1>
      <hr>
      <%
        Gato g1 = new Gato("Pepe", "gato.jpg");
        Gato g2 = new Gato("Garfield", "garfield.jpg");
        Gato g3 = new Gato("Tom", "tom.png");
```

```
        out.println(g1);
        out.println(g2);
        out.println(g3);
        out.println(g1.maulla());
        out.println(g2.come("sardinas"));
      %>
    </body>
</html>
```

12.5 Ejercicios



Ejercicio 1

Crea una aplicación web en Java que muestre tu nombre y tus datos personales por pantalla. La página principal debe ser un archivo con la extensión `jsp`. Comprueba que se lanzan bien el servidor y el navegador web. Observa los mensajes que da el servidor. Fíjate en la dirección que aparece en la barra de direcciones del navegador.



Ejercicio 2

Mejora el programa anterior de tal forma que la apariencia de la página web que muestra el navegador luzca más bonita mediante la utilización de CSS (utiliza siempre ficheros independientes para CSS para no mezclarlo con HTML).



Ejercicio 3

Escribe una aplicación que pida tu nombre. A continuación mostrará "Hola" seguido del nombre introducido. El nombre se deberá recoger mediante un formulario.



Ejercicio 4

Realiza una aplicación que calcule la media de tres notas.



Ejercicio 5

Realiza un conversor de euros a pesetas.



Ejercicio 6

Realiza un conversor de pesetas a euros.



Ejercicio 7

Combina las dos aplicaciones anteriores en una sola de tal forma que en la página principal se pueda elegir pasar de euros a pesetas o de pesetas a euros. Adorna la página con alguna foto o dibujo.



Ejercicio 8

Realiza una aplicación que pida un número y que luego muestre la tabla de multiplicar de ese número. El resultado se debe mostrar en una tabla (`<table>` en HTML).



Ejercicio 9

Realiza una aplicación que pinte una pirámide. La altura se pedirá en un formulario. La pirámide estará hecha de bolitas, ladrillos o cualquier otra imagen.



Ejercicio 10

Realiza un cuestionario con 10 preguntas tipo test sobre las asignaturas que se imparten en el curso. Cada pregunta acertada sumará un punto. El programa mostrará al final la calificación obtenida. Pásale el cuestionario a tus compañeros y pídeles que lo hagan para ver qué tal andan de conocimientos en las diferentes asignaturas del curso. Utiliza *radio buttons* en las preguntas del cuestionario.



Ejercicio 11

Escribe una aplicación que genere el calendario de un mes. Se pedirá el nombre del mes, el año, el texto que queremos que aparezca sobre el calendario, el día de la semana en que cae el día 1 y el número de días que tiene el mes.



Ejercicio 12

Mejora la aplicación anterior de tal forma que no haga falta introducir el día de la semana en que cae el día 1 y el número de días que tiene el mes. El programa debe deducir estos datos del mes y el año. Pista: puedes usar la clase `Calendar` (`java.util.Calendar`).



Ejercicio 13

Transforma el test de infidelidad realizado anteriormente para consola en una aplicación web.



Ejercicio 14

Escribe un programa que muestre los n primeros términos de la serie de Fibonacci. El primer término de la serie de Fibonacci es 0, el segundo es 1 y el resto se calcula sumando los dos anteriores, por lo que tendríamos que los términos son 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144… El número n se debe introducir por teclado.



Ejercicio 15

Realiza una aplicación que genere 100 números aleatorios del 1 al 200. Los primos deberán aparecer en un color diferente al resto.



Ejercicio 16

Realiza una aplicación que muestre la tirada aleatoria de tres dados de póker. Utiliza imágenes de dados reales.



Ejercicio 17

Realiza un configurador del interior de un vehículo. El usuario puede elegir, mediante un formulario, el color de la tapicería – blanco, negro o berenjena - y el material de las molduras – madera o carbono. Se debe mostrar el interior del coche tal y como lo quiere el usuario.



Ejercicio 18

Crea la aplicación “El Trile”. Deben aparecer tres cubiletes por pantalla y el usuario deberá seleccionar uno de ellos. Para dicha selección se puede usar un formulario con radio-button, una lista desplegable, hacer clic en el cubilete o lo que resulte más fácil. Se levantarán los tres cubiletes y se verá si estaba o no la bolita dentro del que el usuario indicó, así mismo, se mostrará un mensaje diciendo “Lo siento, no has acertado” o “¡Enhorabuena!, has encontrado la bolita”. La probabilidad de encontrar la bolita es de 1/3.



Ejercicio 19

Crea el juego “Apuesta y gana”. El usuario debe introducir inicialmente una cantidad de dinero. A continuación aparecerá por pantalla una imagen de forma aleatoria. Si sale una calavera, el usuario pierde todo su dinero y termina el juego. Si sale medio limón, el usuario pierde la mitad del dinero y puede seguir jugando con esa cantidad o puede dejar de jugar. Si sale el gato chino de la suerte, el usuario multiplica por dos su dinero y puede seguir jugando con esa cantidad o puede dejar de jugar.



Ejercicio 20

Crea una aplicación que dibuje un tablero de ajedrez mediante una tabla HTML generada con bucles usando JSP y que sitúe dentro del tablero un alfil y un caballo en posiciones aleatorias. Las dos figuras no pueden estar colocadas en la misma casilla. Las filas y las columnas del tablero deben estar etiquetadas correctamente.



Ejercicio 21

Implementa una máquina de helados. El usuario indica los **porcentajes de helado de chocolate, de fresa y de vainilla**. Los porcentajes deben ser números comprendidos entre 0 y 100. Si los porcentajes suman más de 100, se debe dar un mensaje al usuario que diga **“La suma de los porcentajes debe ser menor o igual que 100. Por favor, introduzca de nuevo los porcentajes”**. En caso de que los porcentajes sean correctos, se mostrará la tarrina con los sabores adecuados y los tamaños bien dimensionados según los porcentajes, tal y como se muestra en los ejemplos. Hay que tener en cuenta que si los sabores no suman el 100% de la tarrina, se debe mostrar el hueco correspondiente. En caso de que no se incluya algún sabor (sabor al 0%), no debe aparecer ninguna referencia a dicho sabor en la tarrina.

Ejemplo 1:

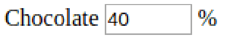
Máquina de helados

Seleccione los porcentajes para preparar un helado a su gusto.









Fresa: 40 %

Vainilla: 40 %

Preparar helado

Máquina de helados

Seleccione los porcentajes para preparar un helado a su gusto.



La suma de porcentajes no pueden ser mayor que el 100%
Por favor, introduzca de nuevo los porcentajes.

Ejemplo 2:

Máquina de helados

Seleccione los porcentajes para preparar un helado a su gusto.



Aquí tiene su tarrina de helado



Ejemplo 3:

Máquina de helados

Seleccione los porcentajes para preparar un helado a su gusto.



Aquí tiene su tarrina de helado





Ejercicio 22

Una empresa que sirve comida vegetariana a domicilio necesita una aplicación para que los clientes puedan hacer sus pedidos por internet. La primera versión incluirá cuatro comidas y tres bebidas. A continuación se muestra una tabla con las diferentes opciones y precios.

Quinoa con verdura	Pizza caprese	Pasta al pesto	Hamburguesa vegetariana	Agua	Cerveza	Refresco
6.95 €	5.50 €	4.90 €	6.20 €	1.00 €	1.50 €	1.40 €

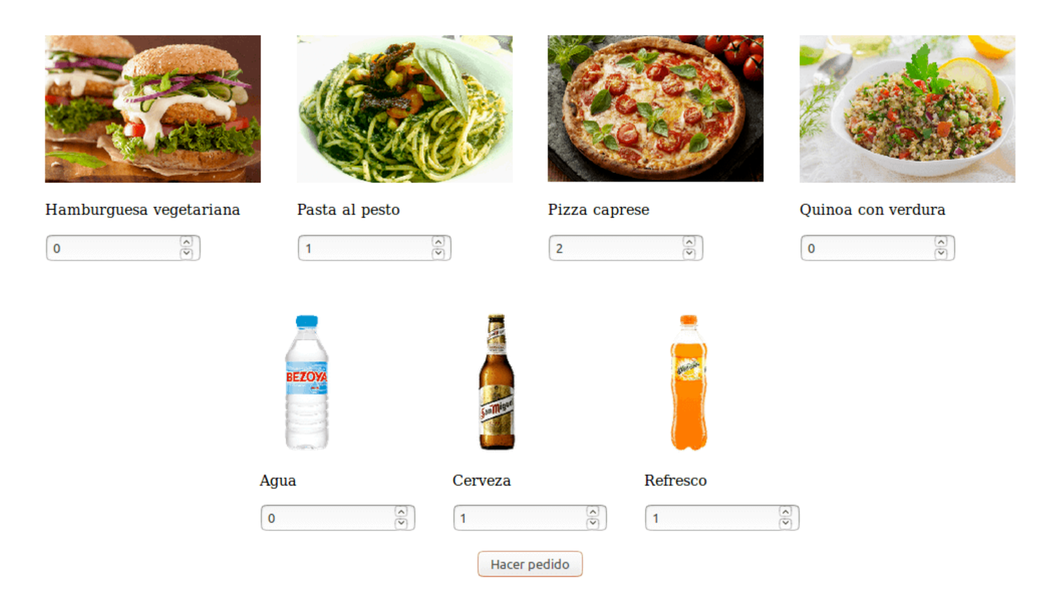
Un pedido puede contener varias comidas del mismo o de distinto tipo y también varias bebidas.

Ejemplos de pedidos:

- 1 quinoa con verdura, 2 hamburguesas vegetarianas, un botellín de agua, una cerveza y un refresco. En total sería 6.95 + 2 x 6.20 + 1 + 1.50 + 1.40 = 23.25 euros.
- 2 pizzas caprese sin bebida. En total sería 2 x 4.90 = 9.80 euros.

Ejemplo:

Formulario de pedido:



Pantalla con el resultado:

Aquí tiene su pedido

Comida / bebida	PVP	Cantidad	Subtotal
Pizza caprese	4.9	2	9.8
Refresco	1.4	1	1.4
Pasta al pesto	5.5	1	5.5
Cerveza	1.5	1	1.5

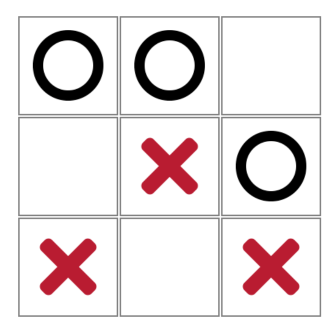
Total 18.2 €



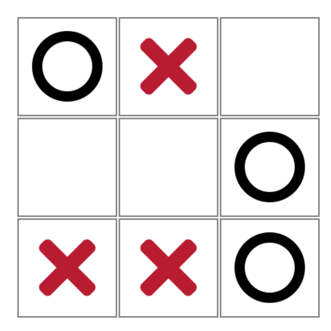
Ejercicio 23

Realiza una aplicación que genere de forma aleatoria una partida finalizada del juego del tres en raya teniendo en cuenta que el tablero tiene tres filas por tres columnas y hay tres círculos y tres cruces que no se pueden solapar. No hay que programar el juego, simplemente mostrar cómo quedaría una partida una vez que ha finalizado.

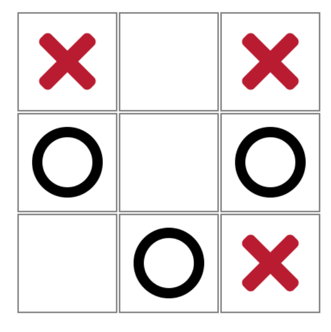
Ejemplo 1:



Ejemplo 2:



Ejemplo 3:



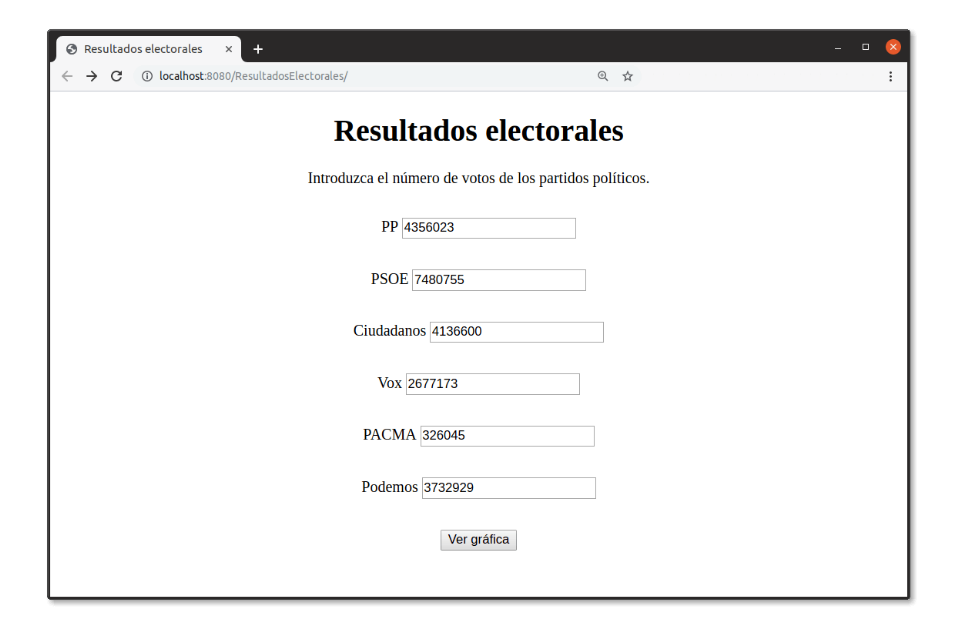


Ejercicio 24

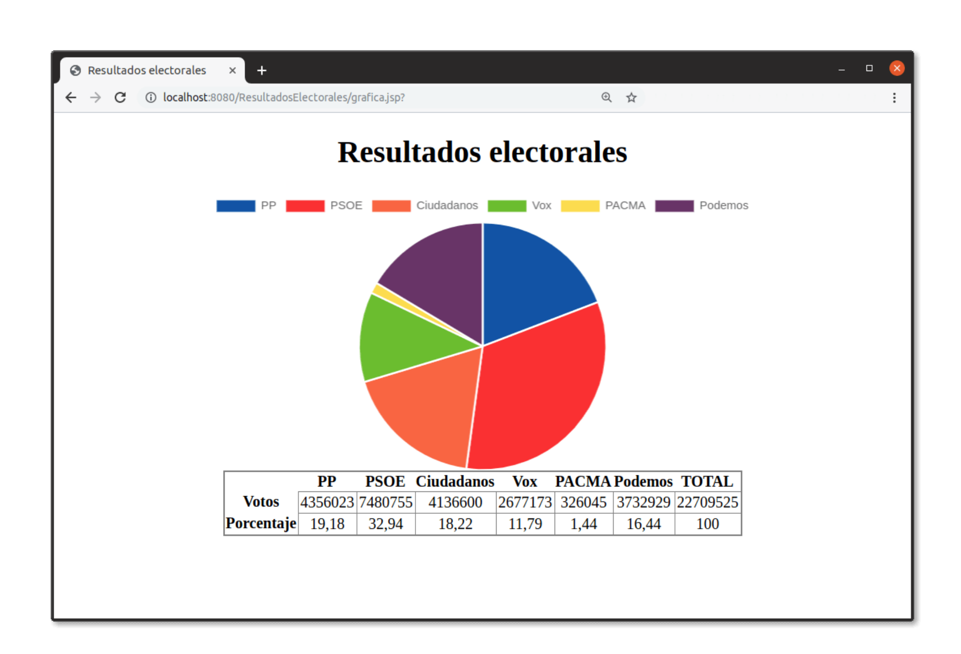
Realiza una aplicación en JSP que recoja mediante un formulario los votos de los diferentes partidos políticos que concurren a las elecciones. A continuación, se debe mostrar una gráfica circular y una tabla con los votos y los porcentajes tal y como se muestra en el ejemplo. Se recomienda usar la librería ChartJs[6]

Ejemplo 1:

[6] https://www.chartjs.org/



Ejemplo 2:





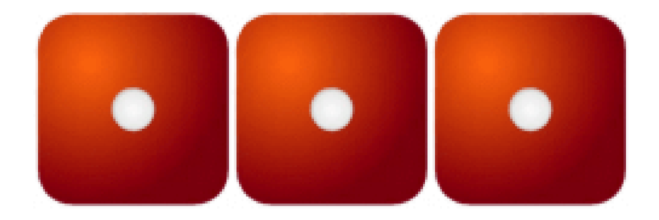
Ejercicio 25

Realiza un programa en JSP que haga tiradas consecutivas de tres dados (en la misma pantalla). El programa parará cuando, en la misma tirada, los tres dados tengan el mismo valor.

Ejemplo 1:



Ejemplo 2:



13. Acceso a bases de datos

En el capítulo 11 vimos cómo almacenar información en ficheros de texto de tal forma que esta información no se pierde cuando el programa se cierra. Normalmente, los ficheros de texto se usan en casos concretos como archivos de configuración o archivos de registro (*log*).

Las bases de datos relacionales entran en juego cuando se necesita almacenar mucha información y, además, esta información es homogénea y se puede representar mediante tablas. Por ejemplo, una aplicación que permita gestionar la matriculación de los alumnos de un centro educativo deberá ser capaz de almacenar los datos de gran cantidad de alumnos (puede que miles). En este caso concreto, cada alumno se representa mediante un **registro** que contiene determinados datos como el número de expediente, el nombre, los apellidos, la nota media, etc. A estos datos se les llama **campos**.

En este capítulo vamos a ver cómo se puede acceder mediante Java a una base de datos relacional **MySQL** para hacer listados, insertar nuevos elementos y borrar elementos existentes.

Los programas de ejemplo que se muestran en este capítulo son aplicaciones JSP, aprovechando lo visto en el capítulo anterior, aunque también se puede acceder a una base de datos desde un programa en Java escrito para la consola.

13.1 Socios de un club de baloncesto

Para ilustrar el acceso a una base de datos mediante Java vamos a utilizar como ejemplo un club de baloncesto. Nuestro club necesita tener almacenada la información de todos los socios. Sobre cada socio se necesita saber su nombre completo, su estatura, su edad y su localidad de residencia. Cada socio tendrá, además, un número de identificación único que será el número de socio.

Nuestra aplicación consistirá en varias páginas JSP que van a permitir ver un listado con los datos de todos los socios, dar de alta un nuevo miembro en el club, y también borrar o modificar los datos de un determinado socio.

Si todavía no tienes instalado en tu equipo el gestor de bases de datos **MySQL**, deberás instalarlo. También es recomendable instalar **PHPMyAdmin** que es una aplicación que permite crear y manejar bases de datos MySQL de una forma muy sencilla, mediante una *interfaz* web.



Instalación de MySQL

```
sudo apt-get install mysql-server
sudo apt-get install mysql-client
```

Para el usuario `root` de MySQL, deberás establecer la contraseña `root` para que te funcionen correctamente los ejemplos.



Instalación de PHPMyAdmin

```
sudo apt-get install phpmyadmin
```

Para acceder a **PHPMyAdmin** debes escribir `http://localhost/phpmyadmin/` en la barra de direcciones del navegador.

Deberás crear la base de datos MySQL a la que llamarás `baloncesto`, que contendrá una tabla de nombre `socio` con los campos `socioID`, `nombre`, `estatura`, `edad` y `localidad` y luego deberás añadir los datos de los socios. Todo esto lo tienes ya preparado en un fichero con nombre `baloncesto.sql` que está en GitHub, en la dirección https://github.com/LuisJoseSanchez/aprende-java-con-ejercicios/blob/master/ejemplos/13_JSP_y_BBDD/baloncesto.sql. Descárgate este fichero, lo utilizaremos en seguida.

A continuación mostramos un fragmento del fichero `baloncesto.sql` para que veas cómo se crea la base de datos, la tabla `socio` y cómo se introducen los datos de muestra.

```
--
-- Base de datos: `baloncesto`
--
CREATE DATABASE IF NOT EXISTS `baloncesto` DEFAULT CHARACTER SET utf8 COLLATE utf8_bin;
USE `baloncesto`;

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `socio`
--

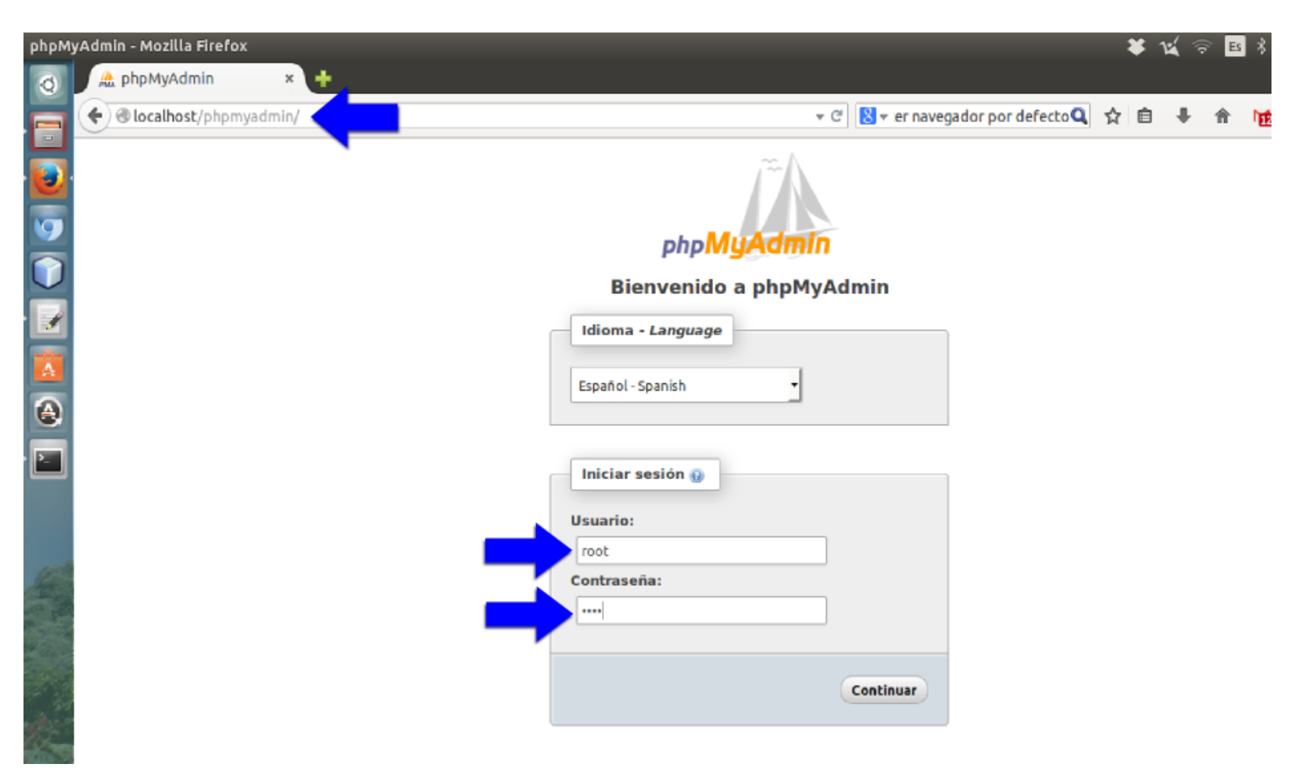
CREATE TABLE IF NOT EXISTS `socio` (
  `socioID` int(11) NOT NULL,
  `nombre` varchar(40) COLLATE utf8_spanish2_ci DEFAULT NULL,
  `estatura` int(11) DEFAULT NULL,
  `edad` int(11) DEFAULT NULL,
  `localidad` varchar(30) COLLATE utf8_spanish2_ci DEFAULT NULL,
  PRIMARY KEY (`socioID`)
```

```
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_spanish2_ci;

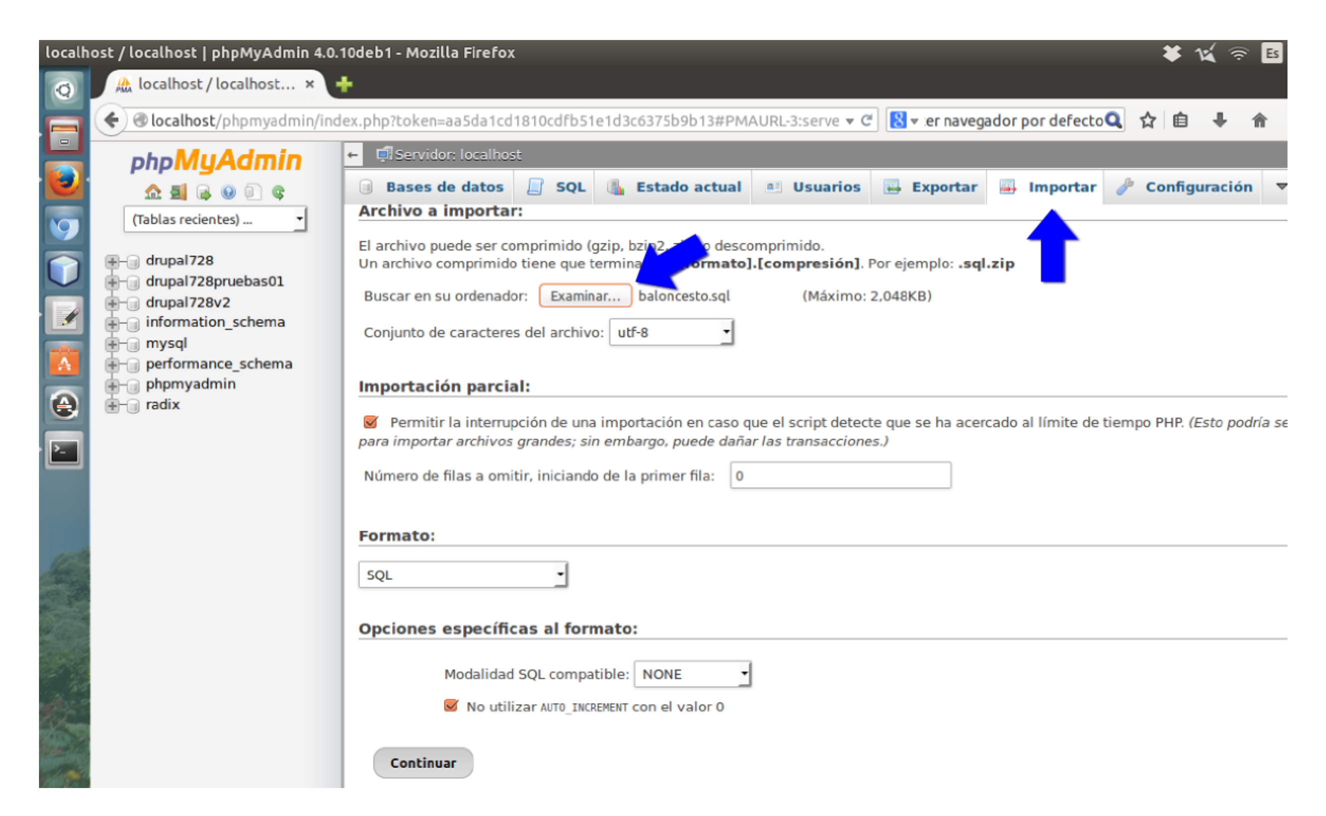
--
-- Volcado de datos para la tabla `socio`
--

INSERT INTO `socio` (`socioID`, `nombre`, `estatura`, `edad`, `localidad`) VALUES
(1235, 'Bermúdez Espada, Ana María', 186, 46, 'Málaga'),
(1236, 'Cano Cuenca, Margarita', 161, 48, 'Málaga'),
...
```

En dos sencillos pasos tendrás preparada y lista para usar la base de datos de ejemplo. Para acceder a PHPMyAdmin debes escribir `http://localhost/phpmyadmin/` en la barra de direcciones de tu navegador. A continuación, teclea el nombre de usuario y la contraseña. Si has instalado MySQL como te hemos indicado, tanto el nombre de usuario como la contraseña deberían ser `root`.



A continuación, haz clic en la pestaña **Importar** y seguidamente en el botón **Examinar**. Selecciona el fichero `baloncesto.sql` que descargaste antes.



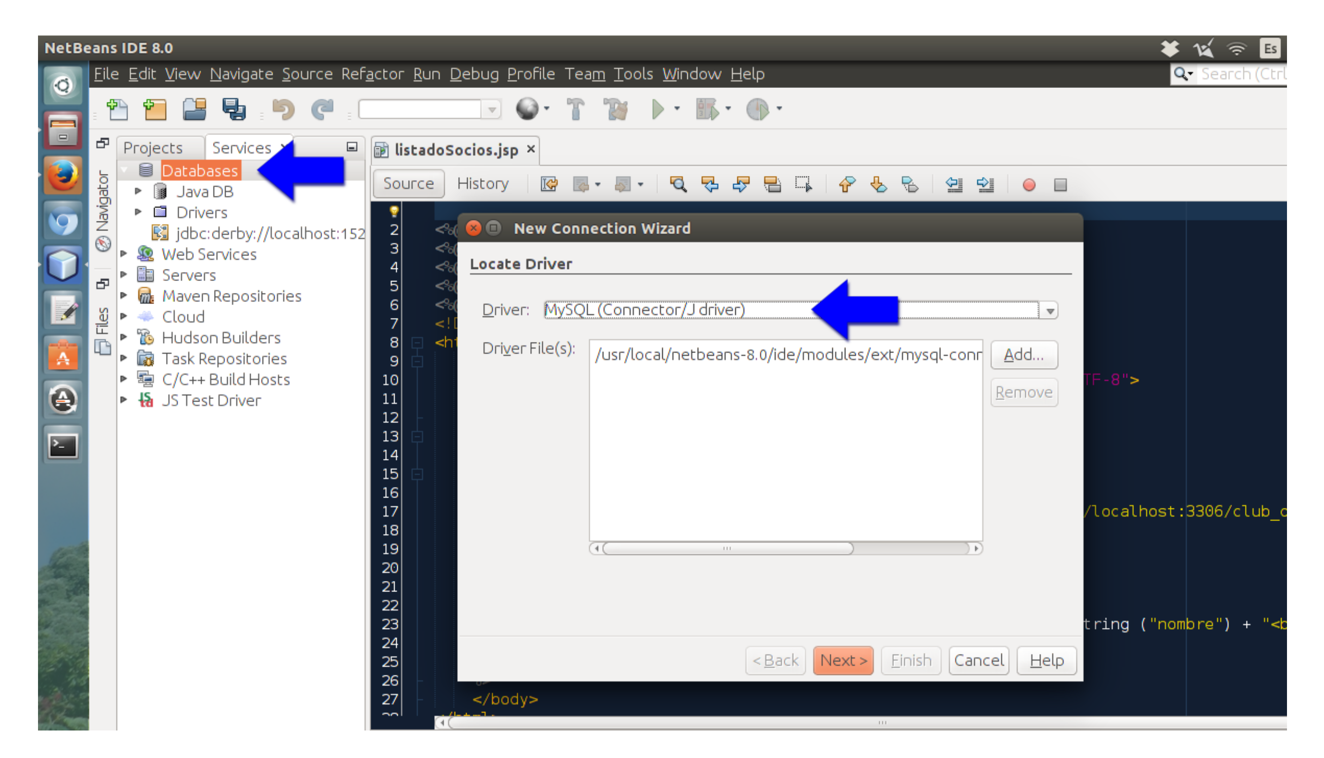
Si has seguido correctamente los pasos, ya tienes la base de datos lista para usar desde un programa escrito en Java.

13.2 Preparación del proyecto de ejemplo

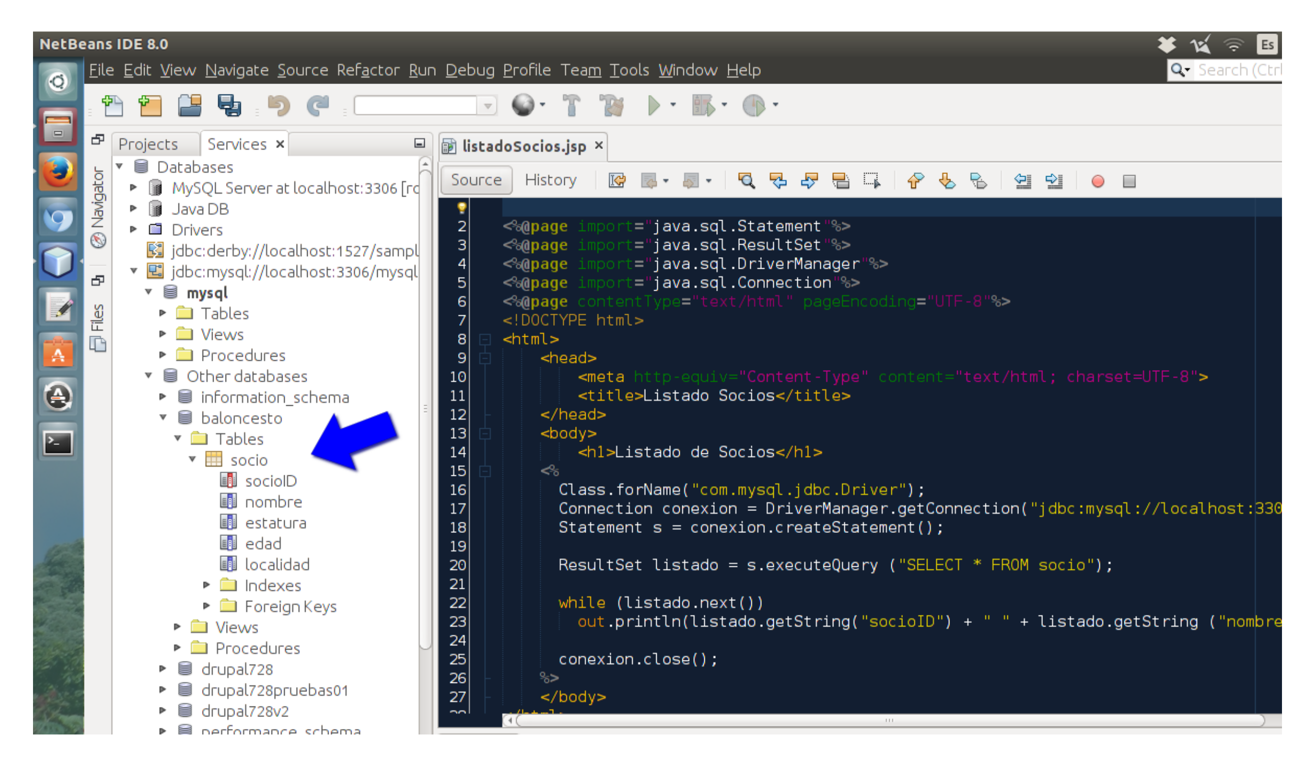
Activar la conexión a MySQL

Abre el entorno Netbeans y crea un nuevo proyecto del tipo **Java Web** - como vimos en el capítulo 12 - y nómbralo **Baloncesto**.

Ahora es necesario activar el servicio de bases de datos. Haz clic en la pestaña **Servicios** (está junto a la pestaña **Proyectos**). Haz clic con el botón derecho en **Bases de datos** y luego selecciona **Nueva conexión**. A continuación aparece una ventana para seleccionar el driver, debes seleccionar **MySQL (Connector/Jdriver)**.



Una vez se haya establecido el servicio, verás las bases de datos disponibles en una estructura de árbol. Puedes ir desplegando hasta encontrar la base de datos `baloncesto` y la tabla `socio`, que contiene los campos `socioID`, `nombre`, `estatura`, `edad` y `localidad`

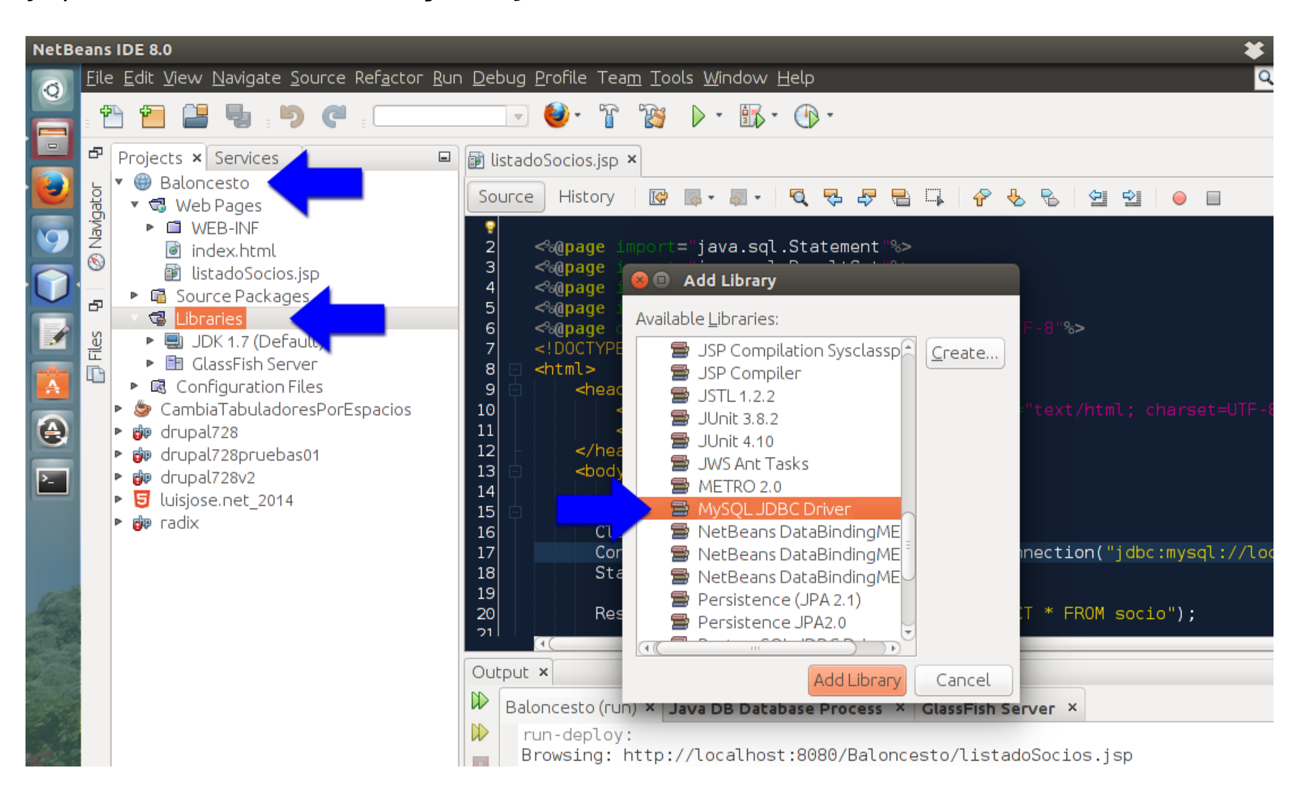


Una vez activada la conexión a MySQL, no es necesario realizarla de nuevo para cada

proyecto.

Incluir la librería MySQL JDBC

Cada proyecto de aplicación en Java que haga uso de una base de datos MySQL deberá incluir la librería **MySQL JDBC**. Para incluir esta librería, despliega el proyecto **Baloncesto**, haz clic con el botón derecho en **Librerías**, selecciona **Añadir librería** y, por último, selecciona **MySQL JDBC Driver**.



13.3 Listado de socios

El primer ejemplo de acceso a una base de datos mediante Java que veremos será un listado. Si echas un vistazo desde PHPMyAdmin a la tabla `socio` de la base de datos `baloncesto` verás que ya contiene información sobre más de 20 socios. Nuestro primer ejemplo será una aplicación en Java que mueste todos esos datos en una página web. Copia el archivo listadoSocios.jsp[1] a la carpeta principal del proyecto (donde está `index.html`). Para ejecutarlo, haz clic con el botón derecho y selecciona **Ejecutar**.

A continuación tienes el código JSP del listado de socios.

[1] https://github.com/LuisJoseSanchez/aprende-java-con-ejercicios/blob/master/ejemplos/13_JSP_y_BBDD/Baloncesto/listadoSocios.jsp

```
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.Connection"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Listado Socios</title>
  </head>
  <body>
    <h1>Listado de Socios</h1>
  <%
    Class.forName("com.mysql.jdbc.Driver");
    Connection conexion = DriverManager.getConnection("jdbc:mysql://localhost:3306/baloncesto"\
,"root", "root");
    Statement s = conexion.createStatement();

    ResultSet listado = s.executeQuery ("SELECT * FROM socio");

    while (listado.next()) {
      out.println(listado.getString("socioID") + " " + listado.getString("nombre") + "<br>");
    }

    conexion.close();
  %>
  </body>
</html>
```

Si todo ha salido bien, al ejecutar el archivo - haciendo clic con el botón derecho sobre el archivo y seleccionando **Ejecutar** - verás por pantalla un listado como el siguiente, con el número y el nombre de cada socio.



Vamos a “diseccionar” el código. Las siguientes tres líneas son obligatorias y deberás copiarlas en todas las páginas JSP que accedan a una base de datos. Lo único que tendrás que cambiar es el nombre de la base de datos - la de nuestro ejemplo se llama `baloncesto` - según el caso.

```
Class.forName("com.mysql.jdbc.Driver");
Connection conexion = DriverManager.getConnection("jdbc:mysql://localhost:3306/baloncesto","ro\
ot", "root");
Statement s = conexion.createStatement();
```

La siguiente línea ejecuta la consulta

```
SELECT * FROM socio
```

sobre la base de datos y guarda el resultado en el objeto `listado`. Esta consulta extrae todos los datos (*) de la tabla `socio`. Si quisiéramos obtener esos datos ordenados por nombre, la consulta sería

```
SELECT * FROM socio ORDER BY nombre
```

Aquí está la línea en cuestión:

```
ResultSet listado = s.executeQuery ("SELECT * FROM socio");
```

Una vez extraídos los datos en bruto, hace falta ir sacando cada uno de los registros, es decir, cada una de las líneas. En cada línea se van a mostrar los datos de un socio. Para ello usamos un `while` que va extrayendo líneas mientras quede alguna en el objeto `listado`.

```
while (listado.next()) {
  out.println(listado.getString("socioID") + " " + listado.getString("nombre") + "<br>");
}
```

Observa que el método `getString` toma como parámetro el nombre de un campo. En este ejemplo estamos mostrando únicamente los números de socio con sus correspondientes nombres. Si quisiéramos mostrar también la altura de cada socio, la podemos extraer mediante `listado.getString("altura")`.

13.4 Alta

Para dar de alta un nuevo socio, necesitamos recoger los datos mediante un formulario que puede ser un programa JSP o simplemente una página HTML. Estos datos recogidos por el formulario serán enviados a otra página encargada de grabarlos en la tabla `socio` de la base de datos `baloncesto` mediante el comando `INSERT` de SQL.

A continuación se muestra el código de `formularioSocio.jsp` que recoge los datos del nuevo socio. Como puedes comprobar es un simple formulario HTML y no contiene código en Java.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  </head>
  <body>
    <h2>Introduzca los datos del nuevo socio:</h2>
    <form method="get" action="grabaSocio.jsp">
      Nº socio <input type="text" name="numero"/></br>
      Nombre <input type="text" name="nombre"/></br>
      Estatura <input type="text" name="estatura"/></br>
      Edad <input type="text" name="edad"/></br>
      Localidad <input type="text" name="localidad"/></br>
      <input type="submit" value="Aceptar">
    </form>
  </body>
</html>
```

El siguiente programa - `grabaSocio.jsp` - se encarga de tomar los datos que envía `formularioSocio.jsp` y grabarlos en la base de datos.

Igual que en los programas del capítulo 12, incluimos esta línea para poder recoger correctamente cadenas de caracteres que contienen tildes, la letra ñ, etc.

```
request.setCharacterEncoding("UTF-8");
```

La variable `insercion` es una cadena de caracteres en la que se va componiendo, a base de ir juntando trozos, la sentencia SQL correspondiente a la inserción.

```
String insercion = "INSERT INTO socio VALUES ("
             + Integer.valueOf(request.getParameter("numero"))
             + ", '" + request.getParameter("nombre")
             + "', " + Integer.valueOf(request.getParameter("estatura"))
             + ", " + Integer.valueOf(request.getParameter("edad"))
             + ", '" + request.getParameter("localidad") + "')";
```

Una vez que se han recogido los datos del formulario y se ha creado la cadena, en la variable `insercion` debería quedar algo como esto:

```
INSERT INTO socio VALUES (6789, "Brito Fino, Alan", 180, 40, "Benalmádena")
```

Por último, se ejecuta la sentencia de inserción mediante `s.execute(insercion)`.

A continuación tienes el código completo de `grabaSocio.jsp`:

```
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.Connection"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  </head>
  <body>
    <%
      Class.forName("com.mysql.jdbc.Driver");
      Connection conexion = DriverManager.getConnection("jdbc:mysql://localhost:3306/baloncest\
o","root", "root");
      Statement s = conexion.createStatement();

      request.setCharacterEncoding("UTF-8");
```

```
      String insercion = "INSERT INTO socio VALUES ("
                        + Integer.valueOf(request.getParameter("numero"))
                        + ", '" + request.getParameter("nombre")
                        + "', " + Integer.valueOf(request.getParameter("estatura"))
                        + ", " + Integer.valueOf(request.getParameter("edad"))
                        + ", '" + request.getParameter("localidad") + "')";
      s.execute(insercion);
      conexion.close();
    %>
    Socio dado de alta.
  </body>
</html>
```



Al componer una sentencia SQL concatenando trozos, es frecuente que se produzcan errores porque faltan o sobran comillas, paréntesis, comas, etc. Si se produce un error al ejecutar la sentencia, la manera más fácil de detectarlo es mostrar por pantalla la sentencia. Por ejemplo, si la sentencia es una inserción de datos y la cadena de caracteres que la contiene se llama `insercion`, como en el ejemplo que acabamos de ver, simplemente tendríamos que escribir: `out.println(insercion)`.

13.5 Borrado

El borrado, al igual que el alta, se realiza en dos pasos. En el primer paso, se carga la página `pideNumeroSocio.jsp` que muestra en una tabla todos los socios, cada uno en una fila. Junto a cada socio se coloca un botón **borrar** que al ser pulsado envía el número de socio a la página `borraSocio.jsp` que se encarga de ejecutar la sentencia SQL de borrado.

La tabla con los socios y los botones de borrado quedaría como se muestra a continuación:

Código	Nombre	Estatura	Edad	Localidad	
1235	Bermúdez Espada, Ana María	186	46	Málaga	borrar
1236	Cano Cuenca, Margarita	161	48	Málaga	borrar
1237	Doña Enríquez, Adrián Manuel	158	31	Málaga	borrar
1238	Fernández Padilla, Esther	183	26	Málaga	borrar
1239	Galán Bazán, Ester María	184	52	Málaga	borrar
1240	Guzmán Puyol, Estefanía	182	30	Málaga	borrar
1241	Martín Jurado, Eva	180	44	Málaga	borrar
1242	Moreno Blanco, Carlos	191	17	Campanillas	borrar
1243	Narváez Gálvez, Juan Antonio	155	22	Campanillas	borrar
1244	Pinto Echeverri, Jhon Diver	167	17	Campanillas	borrar

A continucación se muestra el código de `pideNumeroSocio.jsp`. El fichero estilos.css[2] se encuentra disponible en GitHub, igual que el resto de archivos.

```
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.Connection"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <link rel="stylesheet" type="text/css" href="estilos.css" />
  </head>
  <body>
    <%
      Class.forName("com.mysql.jdbc.Driver");
      Connection conexion = DriverManager.getConnection("jdbc:mysql://localhost:3306/baloncest\
o","root", "root");
      Statement s = conexion.createStatement();
```

[2] https://github.com/LuisJoseSanchez/aprende-java-con-ejercicios/blob/master/ejemplos/13_JSP_y_BBDD/Baloncesto/estilos.css

```
      ResultSet listado = s.executeQuery ("SELECT * FROM socio");
    %>
    <table>
      <tr><th>Código</th><th>Nombre</th><th>Estatura</th><th>Edad</th><th>Localidad</th></tr>
    <%
      while (listado.next()) {
          out.println("<tr><td>");
          out.println(listado.getString("socioID") + "</td>");
          out.println("<td>" + listado.getString("nombre") + "</td>");
          out.println("<td>" + listado.getString("estatura") + "</td>");
          out.println("<td>" + listado.getString("edad") + "</td>");
          out.println("<td>" + listado.getString("localidad") + "</td>");
    %>
      <td>
      <form method="get" action="borraSocio.jsp">
        <input type="hidden" name="codigo" value="<%=listado.getString("socioID") %>"/>
        <input type="submit" value="borrar">
      </form>
      </td></tr>
    <%
      } // while
      conexion.close();
     %>
    </table>
  </body>
</html>
```

A continuación se muestra el código de `borraSocio.jsp`. Como puedes comprobar, la sentencia SQL que borra el registro deseado es un `DELETE` que toma como referencia el número de socio para realizar el borrado.

```
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.Connection"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  </head>
  <body>
    <%
      Class.forName("com.mysql.jdbc.Driver");
      Connection conexion = DriverManager.getConnection("jdbc:mysql://localhost:3306/baloncest\
```

```
o","root", "root");
      Statement s = conexion.createStatement();

      s.execute ("DELETE FROM socio WHERE socioID=" + request.getParameter("codigo"));
    %>
    <script>document.location = "pideNumeroSocio.jsp"</script>
  </body>
</html>
```

13.6 CRUD completo con Bootstrap

Los programas que ofrecen la posibilidad de hacer listados y de realizar modificaciones y borrado sobre los registros de una tabla, se denominan CRUD que son las siglas en inglés de *Create*, *Read*, *Update* y *Delete*; en español se diría que es un programa de alta, listado, modificación y borrado. El código fuente completo está disponible en la carpeta *BaloncestoMejorado* de nuestro repositorio en GitHub[3].

Club de Baloncesto

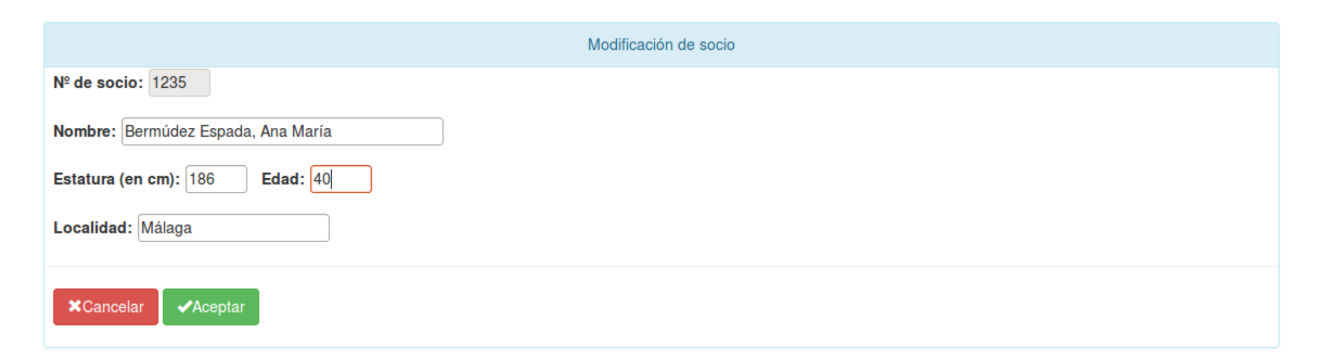
Nº de socio	Nombre	Estatura	Edad	Localidad		
					Añadir	
1235	Bermúdez Espada, Ana María	186	46	Málaga	Modificar	Eliminar
1236	Cano Cuenca, Margarita	161	48	Málaga	Modificar	Eliminar
1237	Doña Enríquez, Adrián Manuel	158	31	Málaga	Modificar	Eliminar
1238	Fernández Padilla, Esther	183	26	Málaga	Modificar	Eliminar
1239	Galán Bazán, Ester María	184	52	Málaga	Modificar	Eliminar
1241	Martín Jurado, Eva	180	44	Málaga	Modificar	Eliminar
1242	Moreno Blanco, Carlos	191	17	Campanillas	Modificar	Eliminar
1243	Narváez Gálvez, Juan Antonio	155	22	Campanillas	Modificar	Eliminar
1244	Pinto Echeverri, Jhon Diver	167	17	Campanillas	Modificar	Eliminar

Para dar de alta un nuevo socio, simplemente habría que rellenar los campos del formulario que aparecen en la primera línea y hacer clic en el botón *Añadir*. La aplicación comprueba si el número de socio introducido ya existe en la base de datos y, en tal caso, muestra un mensaje de error; recuerda que el número de socio es único.

[3]https://github.com/LuisJoseSanchez/aprende-java-con-ejercicios/tree/master/ejemplos/13_JSP_y_BBDD/BaloncestoMejorado

El borrado se lleva a cabo de la misma manera que en el ejemplo del apartado anterior.

El botón *Modificar* nos lleva a otra página que contiene un formulario con todos los datos del socio, donde podemos modificar la información necesaria. A continuación se muestra una captura.



Para los estilos - colores, botones, iconos, etc. - hemos utilizado Boostrap[4]. Boostrap es un **framework** que incluye unos estilos predefinidos para que con solo incluirlo en nuestro proyecto, la aplicación tenga una apariencia bonita y homogénea. Puedes encontrar más información de cómo utilizar este **framework** en la página oficial de Boostrap[5].

[4] http://getbootstrap.com/
[5] http://getbootstrap.com/

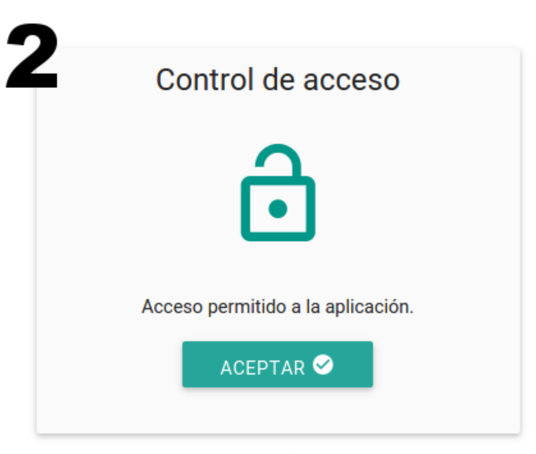
13.7 Ejercicios



Ejercicio 1

Establece un control de acceso mediante nombre de usuario y contraseña para alguno de los programas de la relación anterior (por ejemplo el que pinta una pirámide). Lo primero que aparecerá por pantalla será un formulario pidiendo el nombre de usuario y la contraseña. Si el usuario y la contraseña son correctos, se podrá acceder al ejercicio; en caso contrario, volverá a aparecer el formulario pidiendo los datos de acceso y no se nos dejará ejecutar la aplicación hasta que iniciemos sesión con un nombre de usuario y contraseña correctos. Los nombres de usuario y contraseñas deben estar almacenados en la tabla de una base de datos.





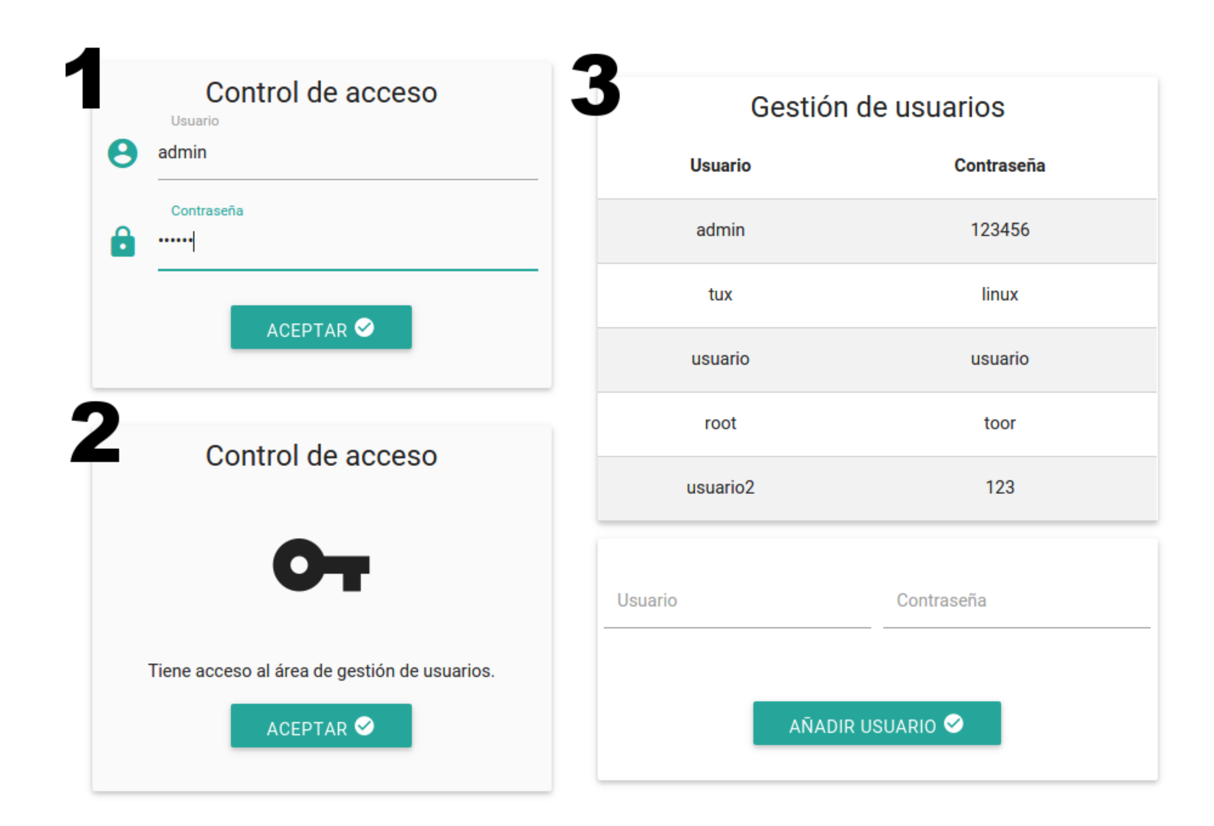






Ejercicio 2

Mejora el programa anterior de tal forma que se puedan dar de alta nuevos usuarios para acceder a la aplicación. Si se introduce un nombre de usuario que no sea el administrador (admin) y una contraseña correcta, la aplicación funcionará exactamente igual que el ejercicio anterior. Si se introduce el usuario `admin` y la contraseña correcta, la aplicación entra en la gestión de usuarios donde se podrán dar de alta nuevos usuarios indicando nombre de usuario y contraseña. No puede haber dos nombres de usuario iguales aunque sí puede haber claves repetidas.





Ejercicio 3

Amplía el programa anterior para que se pueda asignar o quitar permiso para ejecutar las aplicaciones de la relación anterior a los distintos usuarios. Por ejemplo, que se pueda especificar que el usuario "jaimito" pueda ejecutar los ejercicios 2, 3 y 5. Para ello, en la base de datos deberá existir una tabla con las parejas (usuario, nº ejercicio). Por ejemplo, si el usuario "jaimito" tiene acceso a los ejercicios 2, 3 y 5; en la tabla correspondiente estarán las parejas (jaimito, 2), (jaimito, 3) y (jaimito, 5). Lo ideal es que la asignación de permisos se haga mediante el marcado de múltiples "checkbox".

1

Control de acceso

Usuario

admin

Contraseña

••••••

ACEPTAR

2

Control de acceso

Tiene acceso al área de gestión de usuarios.

ACEPTAR

3

Gestión de usuarios

Usuario	Contraseña	Permisos
admin	123456	EDITAR
tux	linux	EDITAR
usuario	usuario	EDITAR
root	toor	EDITAR
usuario2	123	EDITAR

Usuario Contraseña AÑADIR USUARIO

4

Permisos

Usuario	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
tux	✓	✓	☐	☐	✓	✓	☐	☐	☐	✓	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐



Ejercicio 4

Crea una aplicación web que permita hacer listado, alta, baja y modificación sobre la tabla cliente de la base de datos banco. Un cliente tiene su identificador, nombre completo, dirección, teléfono y fecha de nacimiento.

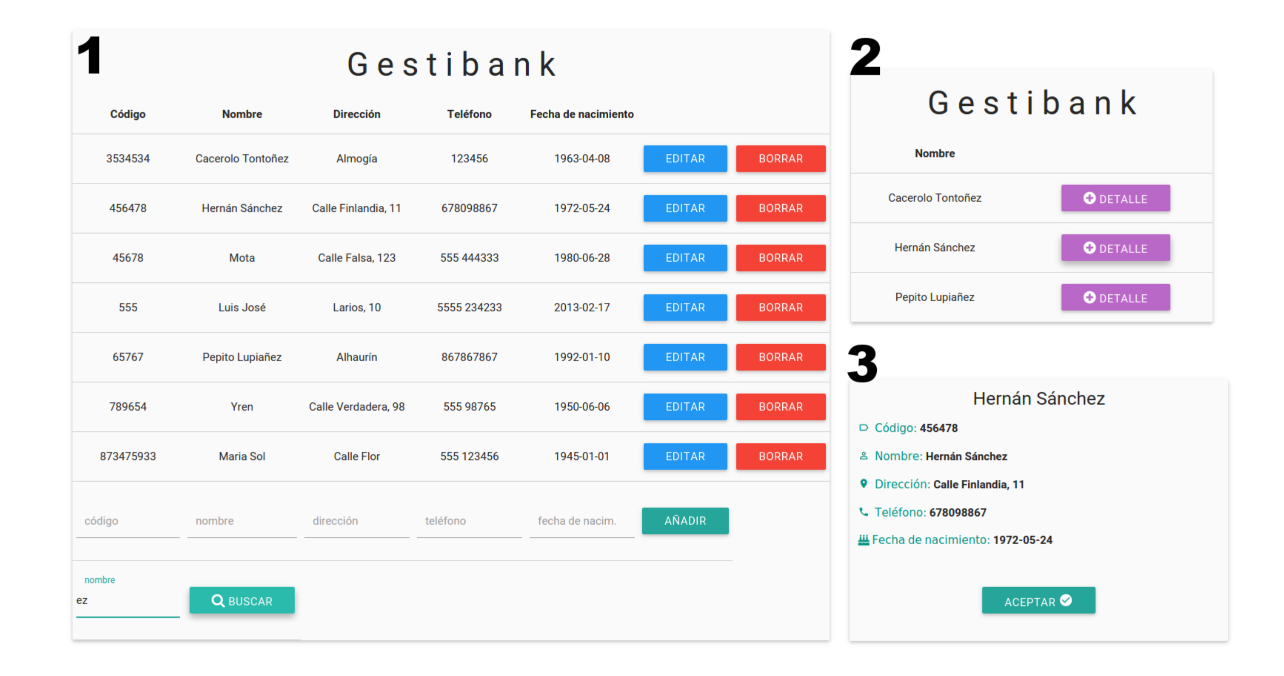
Gestibank

Código	Nombre	Dirección	Teléfono	Fecha de nacimiento		
3534534	Cacerolo Tontoñez	Almogía	123456	1963-04-08	EDITAR	BORRAR
456478	Hernán Sánchez	Calle Finlandia, 11	678098867	1972-05-24	EDITAR	BORRAR
45678	Mota	Calle Falsa, 123	555 444333	1980-06-28	EDITAR	BORRAR
555	Luis José	Larios, 10	5555 234233	2013-02-17	EDITAR	BORRAR
65767	Pepito Lupiañez	Alhaurín	867867867	1992-01-10	EDITAR	BORRAR
76859	ignacio	Periquito, 333	555 325476	1995-08-08	EDITAR	BORRAR
789654	Yren	Calle Verdadera, 98	555 98765	1950-06-06	EDITAR	BORRAR
873475933	Maria Sol	Calle Flor	555 123456	1945-01-01	EDITAR	BORRAR
código	nombre	dirección	teléfono	fecha de nacim.	AÑADIR	



Ejercicio 5

Amplía el programa anterior para que se pueda hacer una búsqueda por nombre. El programa buscará la cadena introducida dentro del campo "nombre" y, si hay varias coincidencias, se mostrará una lista de clientes para poder seleccionar uno de ellos y ver todos los datos. Si solo hay una coincidencia, se mostrarán directamente los datos del cliente en cuestión.





Ejercicio 6

Crea una versión web del programa GESTISIMAL (GESTIón SIMplifcada de Almacén) para llevar el control de los artículos de un almacén. De cada artículo se debe saber el código, la descripción, el precio de compra, el precio de venta y el stock (número de unidades). La entrada y salida de mercancía supone respectivamente el incremento y decremento de stock de un determinado artículo. Hay que controlar que no se pueda sacar más mercancía de la que hay en el almacén. Aprovecha las opciones que puede ofrecer una interfaz web, por ejemplo para eliminar un artículo ya no será necesario pedir el código sino que se puede mostrar un listado de todos los artículos de tal forma que se puedan borrar un artículo directamente pulsando un botón.

Gestisimal

Código	Descripción	Precio de compra	Precio de venta	Stock
2324	MERMELADA DE ALBARICOQUE	1.9	2.3	40
34240	TURRÓN BLANDO ALMENDRA	1.75	2.45	250
36548	PAN DE MOLDE	0.95	1.8	70
4444	CHOCOLATE 150GR	1.9	3.3	5
45654	PAN DE MOLDE	1.78	2.14	23
6767676	TOFU CON SÉSAMO	1.6	2.85	40
67904	CROQUETAS DE SETAS	2.95	3.5	91
89567	PATATAS FRITAS	0.37	0.67	90
código	descripción	precio de compra	precio de venta	stock

14. Control de excepciones

14.1 Errores en tiempo de compilación y en tiempo de ejecución

Programando en Java se pueden producir diferentes tipos de error. El más común es el que se produce en tiempo de compilación y generalmente ocurre cuando hay alguna sentencia que no está bien escrita, por ejemplo si falta un punto y coma al final de una línea, cuando hay comillas o paréntesis que se abren pero no se cierran, al intentar asignar una cadena de caracteres a una variable que es de tipo entero, etc.

Si un programa da errores en tiempo de compilación, no se crea el archivo de *bytecode* con la extensión `.class` y, por tanto, hasta que no se arreglen los fallos, no compilará y, en consecuencia, tampoco se podrá ejecutar.

La siguiente línea de código provocará un error en tiempo de compilación ¿sabrías decir por qué?

```
System.out.println("¡Hola mundo!);
```

Si ya llevas algún tiempo programando en Java, te habrás topado con cientos de errores como éste. Suelen ser los más fáciles de detectar y corregir.



Errores en tiempo de compilación

- Se suelen producir cuando el código no está bien escrito, por ej. cuando falta el punto y coma al final de la sentencia.
- Hasta que no se arreglen, no se genera el archivo con la extensión `.class`

Existen otros errores que se producen en tiempo de ejecución a los que llamaremos **excepciones**. El programa compila y se puede ejecutar pero, por algún motivo, se produce un fallo y el programa se "rompe". Un buen programador debe preveer esta situación y debe saber encauzar el programa para que quede todo bajo control. En este capítulo estudiaremos estos errores y veremos cómo se pueden mantener a raya.

A continuación se muestra un programa que calcula la media de dos números.

```java
import java.util.Scanner;

class EjemploExcepciones01 {
  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);

    System.out.println("Este programa calcula la media de dos números");

    System.out.print("Introduzca el primer número: ");
    double numero1 = Double.parseDouble(s.nextLine());

    System.out.print("Introduzca el segundo número: ");
    double numero2 = Double.parseDouble(s.nextLine());

    System.out.println("La media es " + (numero1 + numero2) / 2);
  }
}
```

El programa compila y se ejecuta correctamente. Entonces ¿dónde está el problema? Prueba a introducir la palabra `hola` en lugar de un número.

```
Este programa calcula la media de dos números
Introduzca el primer número: hola
Exception in thread "main" java.lang.NumberFormatException: For input string: "hola"
        at sun.misc.FloatingDecimal.readJavaFormatString(FloatingDecimal.java:2043)
        at sun.misc.FloatingDecimal.parseDouble(FloatingDecimal.java:110)
        at java.lang.Double.parseDouble(Double.java:538)
        at EjemploExcepciones01.main(EjemploExcepciones01.java:11)
```

El programa ha terminado de forma repentina. ¿Y si, en lugar de ser un simple ejemplo que calcula una media, se tratase de la aplicación que controla un cajero automático o, peor aún, el programa de pilotaje de un avión? Vamos a ver cómo solucionar este problema en los próximos apartados.



Errores en tiempo ejecución

- Se llaman también **excepciones**.
- El programa compila y, en principio, se ejecuta bien… hasta que se dan determinadas circunstancias que provocan un fallo.
- Es necesario preveerlos para que el programa no termine de forma inesperada.

14.2 El bloque `try - catch - finally`

El bloque `try - catch - finally` sirve para encauzar el flujo del programa de tal forma que, si se produce una excepción, no se termine de forma drástica y se pueda reconducir el ejecución de una manera controlada.

El formato de este bloque es el siguiente:

```
try {

  Instrucciones que se pretenden ejecutar
  (si se produce una excepción puede que
   no se ejecuten todas ellas).

} catch {

  Instrucciones que se van a ejecutar
  cuando se produce una excepción.

} finally {

  Instrucciones que se van ejecutar tanto
  si se producen excepciones como si no
  se producen.

}
```

Se pueden especificar varios `catch` para controlar diferentes excepciones como veremos más adelante. La parte `finally` es opcional.

Siguiendo con el programa que calcula la media de dos números, vamos a introducir un bloque `try - catch - finally` para que el programa termine de forma controlada si se produce una excepción.

```java
public class EjemploExcepciones02 {
  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);

    System.out.println("Este programa calcula la media de dos números");

    try {

      System.out.print("Introduzca el primer número: ");
      double numero1 = Double.parseDouble(s.nextLine());
```

```
      System.out.print("Introduzca el segundo número: ");
      double numero2 = Double.parseDouble(s.nextLine());

      System.out.println("La media es " + (numero1 + numero2) / 2);

    } catch (Exception e) {

      System.out.print("No se puede calcular la media. ");
      System.out.println("Los datos introducidos no son correctos.");

    } finally {

      System.out.println("Gracias por utilizar este programa ¡hasta la próxima!");

    }

  }
}
```

Ahora, si el usuario introduce un dato incorrecto, el programa termina de forma ordenada.

```
Este programa calcula la media de dos números
Introduzca el primer número: hola
No se puede calcular la media. Los datos introducidos no son correctos.
Gracias por utilizar este programa ¡hasta la próxima!
```

Se puede mostrar tanto el tipo de excepción como el error exacto que se produce. Para ello, se aplican los métodos `getClass()` y `getMessage()` respectivamente al objeto `e`. El tipo de excepción viene dado por el nombre de una clase que es subclase de `Exception`. Bastará con añadir las siguientes líneas al ejemplo anterior.

```
System.out.println("Excepción: " + e.getClass());
System.out.println("Error: " + e.getMessage());
```

El ejemplo completo quedaría como se indica a continuación.

```java
import java.util.Scanner;

public class EjemploExcepciones02v2 {
  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);

    System.out.println("Este programa calcula la media de dos números");

    try {

      System.out.print("Introduzca el primer número: ");
      double numero1 = Double.parseDouble(s.nextLine());

      System.out.print("Introduzca el segundo número: ");
      double numero2 = Double.parseDouble(s.nextLine());

      System.out.println("La media es " + (numero1 + numero2) / 2);

    } catch (Exception e) {

      System.out.println("Excepción: " + e.getClass());
      System.out.println("Error: " + e.getMessage());
      System.out.println("No se puede calcular la media. ");
      System.out.println("Los datos introducidos no son correctos.");

    }

  }
}
```

Si el usuario introduce un dato incorrecto, ahora el programa muestra la información adicional sobre la excepción que se produce.

```
Este programa calcula la media de dos números
Introduzca el primer número: hola
Excepción: class java.lang.NumberFormatException
Error: For input string: "hola"
No se puede calcular la media.
Los datos introducidos no son correctos.
```

Vamos a refinar un poco más el programa. Si se produce una excepción al introducir un dato, el programa volverá a pedirlo una y otra vez hasta que el dato sea correcto.

```java
import java.util.Scanner;

public class EjemploExcepciones03 {
  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);

    System.out.println("Este programa calcula la media de dos números");

    boolean datoValido = false;

    double numero1 = 0;

    do {
      try {
        System.out.print("Introduzca el primer número: ");
        numero1 = Double.parseDouble(s.nextLine());
        datoValido = true;
      } catch (Exception e) {
        System.out.print("El dato introducido no es correcto, debe ser un número.");
        System.out.println(" Por favor, inténtelo de nuevo.");
      }
    } while (!datoValido);

    double numero2 = 0;

    do {
      try {
        datoValido = false;
        System.out.print("Introduzca el segundo número: ");
        numero2 = Double.parseDouble(s.nextLine());
        datoValido = true;
      } catch (Exception e) {
        System.out.print("El dato introducido no es correcto, debe ser un número.");
        System.out.println(" Por favor, inténtelo de nuevo.");
      }
    } while (!datoValido);

    System.out.println("La media es " + (numero1 + numero2) / 2);

  }
}
```

14.3 Control de varios tipos de excepciones

La clase `Exception` hace referencia a una excepción genérica. Existen muchas subclases de ella como `DataFormatException`, `IOException`, `IndexOutOfBoundsException`, etc.

Mediante la utilización de varios `catch` con diferentes subclases de `Exception` se pueden discriminar distintas excepciones.

Utilizaremos como ejemplo un programa que pide el número total de asteriscos y el número de líneas que se quieren pintar. Por ejemplo, si el usuario dice que quiere pintar 10 asteriscos y 3 líneas, el programa pintará dos líneas de 4 asteriscos más una línea de 2 asteriscos.

```
import java.util.Scanner;

public class EjemploExcepciones04 {
  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);

    System.out.println("Este programa pinta varias líneas de asteriscos");

    System.out.print("Introduzca el número total de asteriscos: ");
    int asteriscos = Integer.parseInt(s.nextLine());

    System.out.print("Introduzca el número de líneas que quiere pintar: ");
    int lineas = Integer.parseInt(s.nextLine());

    int longitud = (asteriscos % lineas) == 0 ? asteriscos / lineas : (int)Math.ceil((double)a\
steriscos / lineas);

    int cuentaAsteriscos = 0;

    for (int i = 1; i <= lineas; i++) {
      System.out.print("Línea " + i + ": ");
      for (int j = 0; (j < longitud) && (cuentaAsteriscos++ < asteriscos); j++) {
        System.out.print("*");
      }
      System.out.println();
    }

  }
}
```

Lo interesante de este ejemplo está en el hecho de que se pueden producir diferentes errores en tiempo de ejecución según los datos que introduzca el usuario.

Cuando el usuario introduce los datos correctamente, es decir, dos números enteros, no hay ningún problema.

```
Este programa pinta varias líneas de asteriscos
Introduzca el número total de asteriscos: 10
Introduzca el número de líneas que quiere pintar: 3
Línea 1: ****
Línea 2: ****
Línea 3: **
```

Ahora bien, si el usuario introduce un número con decimales en lugar de un número entero, se produce la excepción `NumberFormatException` como se muestra a continuación.

```
Este programa pinta varias líneas de asteriscos
Introduzca el número total de asteriscos: 20.75
Exception in thread "main" java.lang.NumberFormatException: For input string: "20.75"
        at java.lang.NumberFormatException.forInputString(NumberFormatException.java:65)
        at java.lang.Integer.parseInt(Integer.java:580)
        at java.lang.Integer.parseInt(Integer.java:615)
        at EjemploExcepciones04.main(EjemploExcepciones04.java:11)
```

La línea que ha producido el fallo se muestra a continuación.

```
int asteriscos = Integer.parseInt(s.nextLine());
```

Lo que ha sucedido es que el método `parseInt()` ha intentado convertir la cadena `"20.75"` en un número entero. Lógicamente, esto no se puede hacer porque un número entero no contiene el punto decimal. Como consecuencia se ha producido la excepción `NumberFormatException`.

Veamos qué otro error se puede producir. Si lo que hace el usuario es introducir el 0 cuando el programa le pregunta cuántas líneas quiere pintar, salta la excepción `ArithmeticException` ya que el programa intenta realizar una división con el divisor igual a 0.

```
Este programa pinta varias líneas de asteriscos
Introduzca el número total de asteriscos: 20
Introduzca el número de líneas que quiere pintar: 0
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at EjemploExcepciones04.main(EjemploExcepciones04.java:16)
```

Ahora, la línea que ha provocado el error es otra distinta.

```
int longitud = (asteriscos % lineas) == 0 ? asteriscos / lineas : (int)Math.ceil((double)aster\
iscos / lineas);
```

Al intentar realizar la división de cualquier número entre la variable `lineas` se produce un error ya que dicha variable vale 0.

Se puede utilizar la estructura `try - catch` con dos excepciones diferentes o más. El siguiente ejemplo es una mejora del anterior en el que se incluye el control de las excepciones `NumberFormatException` y `ArithmeticException`.

```
import java.util.Scanner;

public class EjemploExcepciones05 {
  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);

    System.out.println("Este programa pinta varias líneas de asteriscos");

    try {

      System.out.print("Introduzca el número total de asteriscos: ");
      int asteriscos = Integer.parseInt(s.nextLine());

      System.out.print("Introduzca el número de líneas que quiere pintar: ");
      int lineas = Integer.parseInt(s.nextLine());

      int longitud = (asteriscos % lineas) == 0 ? asteriscos / lineas : (int) Math.ceil((doubl\
e) asteriscos / lineas);

      int cuentaAsteriscos = 0;

      for (int i = 1; i <= lineas; i++) {
        System.out.print("Línea " + i + ": ");
        for (int j = 0; (j < longitud) && (cuentaAsteriscos++ < asteriscos); j++) {
          System.out.print("*");
        }
        System.out.println();
      }

    } catch (NumberFormatException nfe) {

      System.out.println("Los datos introducidos no son correctos.");
      System.out.println("Se deben introducir números enteros.");

    } catch (ArithmeticException ae) {
```

```
      System.out.println("El número de líneas no puede ser 0.");

    }

  }
}
```

Ahora, si el usuario introduce un número con decimales o una palabra en lugar de un número entero, no hay ningún problema, el programa no se rompe y termina de forma ordenada.

```
Este programa pinta varias líneas de asteriscos
Introduzca el número total de asteriscos: 10
Introduzca el número de líneas que quiere pintar: 4.6
Los datos introducidos no son correctos.
Se deben introducir números enteros.
```

De igual modo, el programa también controla la excepción que se produce si el usuario introduce un 0 para indicar el número de líneas que quiere pintar.

```
Este programa pinta varias líneas de asteriscos
Introduzca el número total de asteriscos: 10
Introduzca el número de líneas que quiere pintar: 0
El número de líneas no puede ser 0.
```

14.4 Lanzamiento de excepciones con `throw`

La orden `throw` permite lanzar de forma explícita una excepción. Por ejemplo, la sentencia `throw new ArithmeticException()` crea de forma artificial una excepción igual que si existiera una línea como `System.out.println(1 / 0);`.

```
public class EjemploExcepciones06throw {
  public static void main(String[] args) {
    System.out.println("Inicio");
    throw new ArithmeticException();
  }
}
```

Este programa provoca la siguiente salida.

```
Inicio
Exception in thread "main" java.lang.ArithmeticException
        at EjemploExcepciones06throw.main(EjemploExcepciones06throw.java:5)
```

De la misma forma, el programa que se muestra a continuación genera también una excepción del tipo `ArithmeticException`.

```
public class EjemploExcepciones07sinthrow {
  public static void main(String[] args) {
    System.out.println("Inicio");
    System.out.println(1 / 0);
  }
}
```

```
Inicio
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at EjemploExcepciones07sinthrow.main(EjemploExcepciones07sinthrow.java:5)
```

Hay que tener en cuenta que `throw` solo puede lanzar excepciones que pertenezcan a la clase `Throwable`.

Como `throw` permite lanzar de forma explícita una excepción, nos servirá para lanzar excepciones propias como veremos más adelante. También es útil cuando se recoge la excepción en un método y luego, esa misma excepción se vuelve a lanzar para que la recoja, a su vez, otro método y luego otro y así sucesivamente hasta llegar al `main`.

Vamos a ver cómo se recoge y se trata una excepción dentro de una función y, además es la propia función la que *"pasa la bola"* al main. Partimos de un ejemplo sin control de excepciones.

```
import java.util.Scanner;

public class EjemploExcepciones08manzanas {
  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    System.out.print("Número de manzanas: ");
    int m = Integer.parseInt(s.nextLine());
    System.out.print("Número de personas: ");
    int p = Integer.parseInt(s.nextLine());
    System.out.print("A cada persona le corresponden " + reparteManzanas(m, p) + " manzanas.");
  }

  public static int reparteManzanas(int manzanas, int personas) {
    return manzanas / personas;
  }
}
```

Cuando el usuario introduce los datos correctamente, el programa realiza su cometido.

```
Número de manzanas: 10
Número de personas: 5
A cada persona le corresponden 2 manzanas.
```

Pero cuando el usuario introduce un 0 como número de personas, el programa provoca una excepción.

```
Número de manzanas: 10
Número de personas: 0
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at EjemploExcepciones08manzanas.reparteManzanas(EjemploExcepciones08manzanas.java:15)
        at EjemploExcepciones08manzanas.main(EjemploExcepciones08manzanas.java:11)
```

Vamos a mejorar el ejemplo para llevar un control de las excepciones tanto en la función como en el `main`.

```java
import java.util.Scanner;

public class EjemploExcepciones09ManzanasConThrow {
  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    System.out.print("Número de manzanas: ");
    int m = Integer.parseInt(s.nextLine());
    System.out.print("Número de personas: ");
    int p = Integer.parseInt(s.nextLine());
    try {
      System.out.print("A cada persona le corresponden " + reparteManzanas(m, p) + " manzanas.\
");
    } catch (ArithmeticException ae) {
      System.out.println("Los datos introducidos no son correctos.");
    }
  }

  public static int reparteManzanas(int manzanas, int personas) {
    try {
      return manzanas / personas;
    } catch (ArithmeticException ae) {
      System.out.println("El número de personas vale 0.");
      throw ae;
    }
  }
}
```

```
Número de manzanas: 10
Número de personas: 0
El número de personas vale 0.
Los datos introducidos no son correctos.
```

14.5 Declaración de un método con `throws`

Hemos visto en el apartado anterior cómo lanzar una excepción desde una función/método con `throw`. Es muy recomendable indicar de forma explícita en la cabecera que existe esa posibilidad, es decir, que el método en cuestión puede provocar una excepción. Se trata de una declaración de intenciones, algo parecido al `@Override`. Si el IDE que estemos utilizando detecta que un método tiene `throws` en su cabecera y, sin embargo, no hemos gestionado bien la posibilidad de provocar una excepción, nos avisará con el correspondiente mensaje de error.

Vamos a modificar la cabecera de la función `reparteManzanas()` para incluir `throws`.

```
public static int reparteManzanas(int manzanas, int personas) throws ArithmeticException
```

El código completo quedaría como sigue.

```
import java.util.Scanner;

public class EjemploExcepciones09ManzanasConThrowYThrows {
  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    System.out.print("Número de manzanas: ");
    int m = Integer.parseInt(s.nextLine());
    System.out.print("Número de personas: ");
    int p = Integer.parseInt(s.nextLine());
    try {
      System.out.print("A cada persona le corresponden " + reparteManzanas(m, p) + " manzanas.\
");
    } catch (ArithmeticException ae) {
      System.out.println("Los datos introducidos no son correctos.");
    }
  }

  public static int reparteManzanas(int manzanas, int personas) throws ArithmeticException {
    try {
      return manzanas / personas;
    } catch (ArithmeticException ae) {
      System.out.println("El número de personas vale 0.");
```

```
      throw ae;
    }
  }
}
```

14.6 Creación de excepciones propias

Java nos permite crear excepciones propias, hechas a medida. Para ello, no hay más que utilizar una de las características más importantes de la programación orientada a objetos: la herencia. Crear una nueva excepción será tan sencillo como implementar una subclase de `Exception`.

Veamos paso a paso cómo crear y utilizar una excepción propia.

Tenemos un programa que pinta por pantalla una pirámide, tras haber pedido la altura a un usuario. En principio, el contenido del `main` sería el que se muestra a continuación.

```
Scanner s = new Scanner(System.in);

System.out.print("Introduzca la altura de la pirámide (un número entre 1 y 10): ");
int h = Integer.parseInt(s.nextLine());

pintaPiramide(h);
```

El código de la función `pintaPiramide` se muestra a continuación.

```
  public static void pintaPiramide(int h) {

    int planta = 1;
    int longitudDeLinea = 1;
    int espacios = h - 1;

    while (planta <= h) {

      // inserta espacios
      for (int i = 1; i <= espacios; i++) {
        System.out.print(" ");
      }

      // pinta la línea
      for (int i = 1; i <= longitudDeLinea; i++) {
        System.out.print("*");
      }
```

```
    System.out.println();

    planta++;
    espacios--;
    longitudDeLinea += 2;
  }

}
```

Ahora bien, vamos a crear una nueva excepción llamada `ExcepcionAlturaFueraDeRango` de tal forma que si alguien intenta pintar una pirámide con una altura menor que 1 o mayor que 10, salte el error y se pueda tratar con un bloque `try - catch`. Será algo parecido al `IndexOutOfBoundsException` que se produce cuando intentamos acceder a una posición que no existe dentro de un array.

Crearemos una clase que se llame precisamente `ExcepcionAlturaFueraDeRango`.

public class ExcepcionAlturaFueraDeRango extends Exception { public ExcepcionAlturaFueraDeRango() { System.out.println("ExcepcionAlturaFueraDeRango: La altura está fuera del rango permitido."); } }

Así de fácil, ya tenemos nuestra excepción personalizada. Veamos ahora cómo utilizarla desde nuestra función `pintaPirámide` y desde el programa principal.

Si queremos que la función `pintaPirámide` pueda arrojar la excepción `ExcepcionAlturaFueraDeRango`, lo tendremos que indicar en la cabecera.

```
public static void pintaPiramide(int h) throws ExcepcionAlturaFueraDeRango
```

Dentro de la función, se comprueba si la altura pasada como parámetro está fuera del rango permitido. En tal caso, se lanza la excepción con `throw`.

```
if ((h < 1) || (h > 10)) {
  throw new ExcepcionAlturaFueraDeRango();
}
```

Nos queda tratar la posible excepción desde el programa principal mediante un bloque `try-catch`.

```
try {
  pintaPiramide(h);
} catch (ExcepcionAlturaFueraDeRango eafr) {
  System.out.println("No se ha podido pintar la pirámide.");
}
```

Juntando todas las piezas, el programa completo quedaría como se muestra a continuación.

```java
import java.util.Scanner;

public class PruebaExcepcionesPropias {
  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);

    System.out.print("Introduzca la altura de la pirámide (un número entre 1 y 10): ");
    int h = Integer.parseInt(s.nextLine());

    try {
      pintaPiramide(h);
    } catch (ExcepcionAlturaFueraDeRango eafr) {
      System.out.println("No se ha podido pintar la pirámide.");
    }
  }

  public static void pintaPiramide(int h) throws ExcepcionAlturaFueraDeRango {

    if ((h < 1) || (h > 10)) {
      throw new ExcepcionAlturaFueraDeRango();
    }

    int planta = 1;
    int longitudDeLinea = 1;
    int espacios = h - 1;

    while (planta <= h) {

      // inserta espacios
      for (int i = 1; i <= espacios; i++) {
        System.out.print(" ");
      }

      // pinta la línea
      for (int i = 1; i <= longitudDeLinea; i++) {
        System.out.print("*");
      }

      System.out.println();

      planta++;
      espacios--;
      longitudDeLinea += 2;
    }

  }
```

```
}
```

14.7 Recomendaciones sobre el tratamiento de excepciones

En este capítulo hemos visto cómo tratar diferentes tipos de excepciones para prevenir que un programa se rompa y termine de forma abrupta. No obstante, en términos generales, no se debe dejar en manos del control de excepciones lo que se puede hacer mediante sencillas comprobaciones previas.

Por ejemplo, antes de realizar una división, el programador puede utilizar un `if` para comprobar si el divisor es 0 y, en tal caso, derivar el flujo de ejecución convenientemente sin tener que recurrir a las excepciones.

Sucede algo muy parecido con la excepción `IndexOutOfBounds`. Si un programador maneja bien los índices de un array, este error nunca debería llegar a producirse.

14.8 Ejercicios



Ejercicio 1

Realiza un programa que pida 6 números por teclado y nos diga cuál es el máximo. Si el usuario introduce un dato erróneo (algo que no sea un número entero) el programa debe indicarlo y debe pedir de nuevo el número.



Ejercicio 2

Modifica la clase `Gato` vista anteriormente añadiendo el método `apareaCon`. Este método debe comprobar que los gatos son de distinto sexo, tras lo cual, genera un nuevo gato. Por ejemplo, sería válido algo como `Gato cria = garfield.apareaCon(lisa)`. En caso de que los gatos sean del mismo sexo, el método debe generar la excepción `ExcepcionApareamientoImposible`. Crea un programa desde el que se pruebe el método.



Ejercicio 3

Realiza un programa que genere una excepción de forma aleatoria de entre las siguientes excepciones: `NumberFormatException`, `IOException`, `FileNotFoundException`, `IndexOutOfBoundsException` y `ArithmeticException`.

15. Sesiones y *cookies*

15.1 Sesiones

En las aplicaciones JSP, las sesiones se utilizan para guardar información en memoria, de tal forma que esa información no se pierde si el usuario navega por la aplicación saltando de una página a otra.

Hemos visto anteriormente cómo enviar datos entre dos páginas mediante un formulario y también mediante un enlace, ahora imagina que visitas varias veces la misma página, que saltas a otra, que vuelves de nuevo a la primera; sería muy engorroso estar mandando datos constantemente mediante formularios o enlaces entre unas páginas y otras. Mediante las sesiones se puede conservar o modificar la información que nos interese independientemente de la/s página/s que se vayan visitando. El valor que se almacena en la memoria está disponible mientras no se cierre la sesión.

Un uso típico de una sesión es el carrito de la compra de una tienda on-line. Podemos visitar todas las páginas que queramos de la tienda e ir añadiendo o quitando productos del carrito gracias a que esta información se graba en una sesión.

Principales métodos para el manejo de sesiones

A continuación se describen los métodos que permiten manejar las sesiones en JSP. Se muestra la cabecera[1] y la acción que realiza cada método.

void setAttribute(String name, Object value)

Guarda un objeto en la sesión con el nombre especificado. El nombre es de tipo `String`, por tanto, no hay que olvidar entrecomillarlo. El objeto puede ser de cualquier tipo, puede ser otra cadena de caracteres o un entero o incluso una lista.

Ejemplos:

```
session.setAttribute("color", "verde")

session.setAttribute("color", request.getParameter("inputcolor"))

session.setAttribute("numero", 24)

session.setAttribute("aleatorio", (int)(Math.random() * 10) + 1)
```

[1] Extraído de la documentación oficial sobre el interface `HttpSession` https://docs.oracle.com/javaee/7/api/javax/servlet/http/HttpSession.html.

Object getAttribute(String name)

Devuelve el objeto guardado en la sesión bajo el nombre especificado. En ocasiones será necesario hacerle un *casting* al objeto devuelto para tener el tipo deseado. Si no existe una variable de sesión con el nombre especificado, se devuelve `null`. Los métodos `getAttribute()` y `setAttribute()` son los más usados con diferencia cuando se trabaja con sesiones.

Ejemplos:

```
El color es <%= session.getAttribute("color") %>

String miColor = (String)session.getAttribute("color")

int n = (Integer)session.getAttribute("aleatorio")
```

void removeAttribute(String name)

Borra un objeto de la sesión. Si se intenta recuperar con `getAttribute()` una vez borrado, se obtiene el valor `null`.

Ejemplos:

```
session.removeAttribute("color")

session.removeAttribute("aleatorio")
```

Enumeration<String> getAttributeNames()

Devuelve los nombres de todos los objetos de la sesión. Nótese que se devuelve en forma de enumerado de cadenas de caracteres; por tanto, para recorrerlo con un `for` hay que convertirlo en una lista mediante `Collections.list(session.getAttributeNames())`.

Ejemplo:

```
for(String nombre : Collections.list(session.getAttributeNames())) {

 out.print("<br>Nombre: " + nombre + ", Valor: " + session.getAttribute(nombre));

}
```

long getCreationTime()

Devuelve el momento en que se creó la sesión. Se cuentan los milisegundos transcurridos desde las cero horas del uno de enero de 1970. Para pasarlo a una fecha y hora legibles se puede usar la clase `Calendar` tal como se muestra en el ejemplo.

Ejemplo:

```
DateFormat formato = new SimpleDateFormat("dd/MM/yyyy hh:mm:ss");

Calendar calendar = Calendar.getInstance();

calendar.setTimeInMillis(session.getCreationTime());

out.println("Incio de sesión: " + formato.format(calendar.getTime()));
```

String getId()

Devuelve el identificador asociado a la sesión.

Ejemplo:

```
String id = session.getId()
```

boolean isNew()

Devuelve `true` si la sesión es nueva, es decir, si se acaba de crear y `false` en caso contrario, o sea, si la sesión ya estaba creada.

Ejemplo:

```
// Muestra true cuando se arranca la aplicación

out.println(session.isNew());

// Pulsa F5 para recargar la página y comprueba que ahora se muestra false
```

void invalidate()

Cierra la sesión y, por tanto, destruye todos los objetos que estaban vinculados a ella.

Ejemplo:

```
session.invalidate()
```

void setMaxInactiveInterval(int interval)

Especifica el tiempo en segundos que una sesión permanecerá abierta desde el momento en que el cliente permanezca inactivo, es decir, desde que no acceda a ninguna variable de sesión.

Si se especifica un cero o un número negativo, no habrá ninguna restricción de tiempo y la sesión expirará cuando el cliente cierre el navegador.

Ejemplo:

```
// Cierra la sesión si el cliente no hace peticiones durante 5 minutos

session.setMaxInactiveInterval(300);
```

Asignación y recuperación de objetos de una sesión

Veamos un ejemplo sencillo en el que se asignan y se recuperan objetos de una sesión.

En la página principal `index.jsp` se asignan una serie de valores que se van a conservar en memoria. Esta página tiene dos enlaces que van a `pagina1.jsp` y `pagina2.jsp` respectivamente. Observa que no se pasan datos por ningún formulario ni tampoco mediante los enlaces.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Sesiones</title>
  </head>
  <body>
    <%
      session.setAttribute("palabra", "hola");
      session.setAttribute("aleatorio", (int) (Math.random() * 10) + 1);
      session.setAttribute("color", "verde");
    %>
    <p>Variables de sesión asignadas.</p>
    <p><a href="pagina1.jsp">Página 1</a></p>
    <p><a href="pagina2.jsp">Página 2</a></p>
  </body>
</html>
```

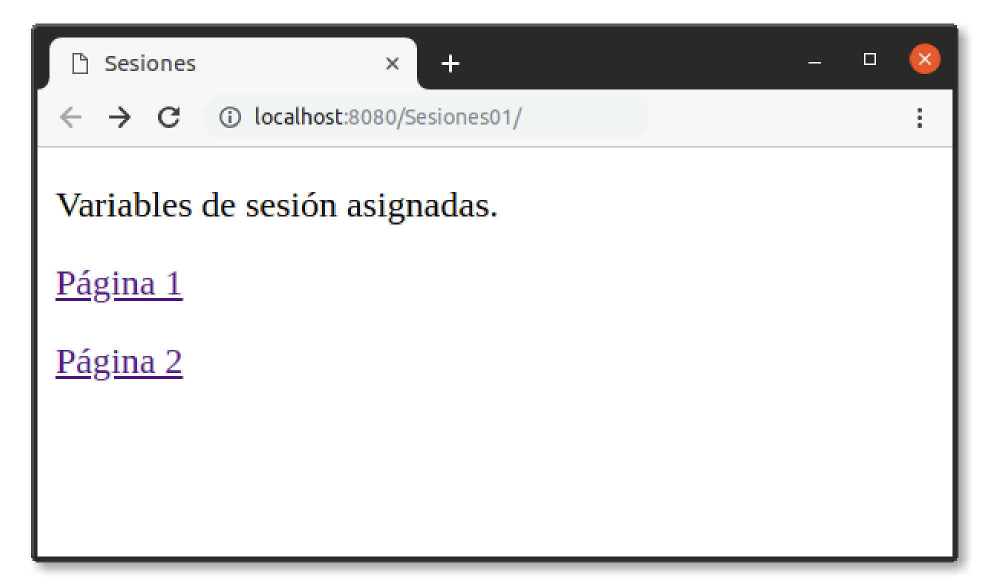


Figura 15.1: Carga las variables de sesión.

En la **página 1** se muestra el contenido de todas las variables de sesión. Para ello se hace uso del método `getAttributeNames()`.

```
<%@page import="java.util.Collections"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <style>
      table, th, td {
        border: 1px solid black;
        border-collapse: collapse;
      }
    </style>
    <title>Sesiones</title>
  </head>
  <body>
    <table>
      <tr><th>Nombre</th><th>Valor</th></tr>
      <%
        for (String nombre : Collections.list(session.getAttributeNames())) {
          out.print("<tr><td>" + nombre + "</td>");
          out.print("<td>" + session.getAttribute(nombre) + "</td></tr>");
        }
      %>
    </table>
  </body>
</html>
```



Figura 15.2: Muestra todos los objetos que contiene la sesión.

En la **página 2** se utiliza el método `getAttribute()` para obtener el valor de un objeto asociado a la sesión. Recuerda que si intentamos recuperar un objeto que no existe, recibiremos `null` como respuesta.

```
<%@page import="java.util.Calendar"%>
<%@page import="java.text.SimpleDateFormat"%>
<%@page import="java.text.DateFormat"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Sesiones</title>
  </head>
  <body>
    <%
      String miColor = (String) session.getAttribute("color");
      int n = (Integer) session.getAttribute("aleatorio");
      out.println("<p>El color es <b>" + miColor + "</b>");
      out.print("y el número es <b>" + n + "</b></p>");
    %>
    <p>Otra variable vale <b><%= session.getAttribute("otra")%></b></p>
    <p>
      <%
        DateFormat formato = new SimpleDateFormat("dd/MM/yyyy hh:mm:ss");
        Calendar calendar = Calendar.getInstance();
        calendar.setTimeInMillis(session.getCreationTime());
        out.println("Incio de sesión: <b>" + formato.format(calendar.getTime()) + "</b>");
      %>
    </p>
    <p>Id de la sesión: <b><%= session.getId() %></b></p>
  </body>
</html>
```

También se utilizan los métodos `getCreationTime()` y `getId()` que se emplean para obtener el momento (fecha y hora) en que se inició la sesión y para obtener el identificador de la sesión respectivamente.

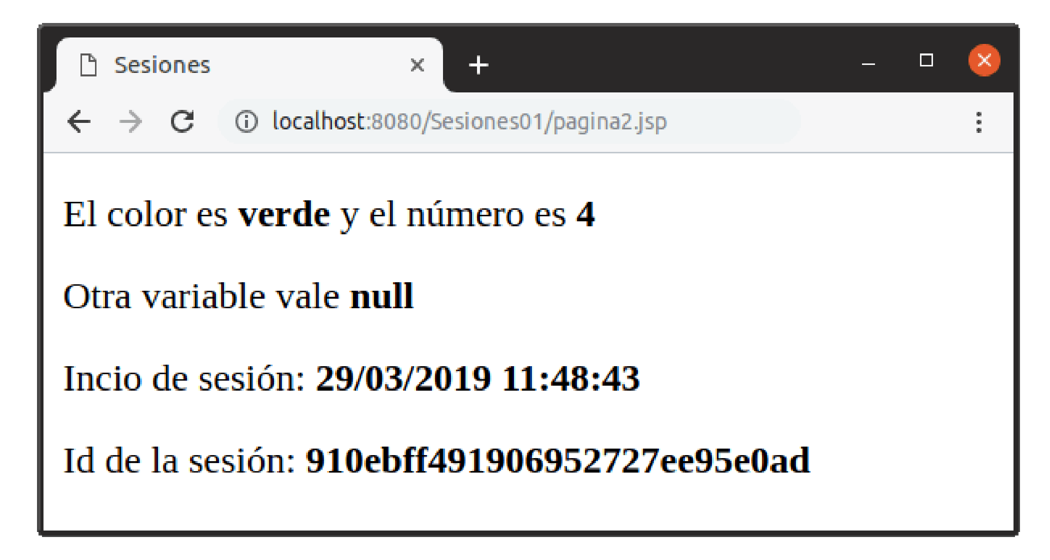


Figura 15.3: Recupera valores, obtiene fecha y hora de inicio y obtiene identificador de la sesión.



Los métodos `setAttribute()` y `getAttribute()` son, con mucha diferencia, los más utilizados en el manejo de sesiones en Java.

Comprobación de sesión nueva

Puede resultar útil en ocasiones comprobar si la sesión es nueva, es decir, si acaba de ser creada. Para ello se utiliza el método `isNew()`. En el siguiente ejemplo podemos ver en funcionamiento este método.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Sesión nueva</title>
    </head>
    <body>
        Sesión nueva: <%= session.isNew() %><br>
        Recarga la página con F5
    </body>
</html>
```

Ejecútalo y observa cómo la primera vez que se carga la página se muestra `true`. Si recargas la página comprobarás que se muestra `false`.



El objeto `session` se crea por defecto. No es necesario crearlo de forma explícita. Tampoco es necesario indicar en cada página JSP que se deben mantener en memoria los valores, se hace todo de forma automática.

Contador de visitas con sesiones

Uno de los ejemplos más sencillos del uso de sesiones es un contador de visitas. Se trata de un contador de visitas "local" ya que sirve para saber el número de accesos que realiza un determinado cliente (navegador); cada persona que se conecta a la página, tiene su propia cuenta.

La primera vez que se carga la página, `session.getAttribute("n")` vale `null`, por tanto, se le asigna el valor 1.

Cuando `session.getAttribute("n")` tiene un valor distinto de `null`, se incrementa en 1.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Contador de visitas</title>
  </head>
  <body>
    <%
      session.setAttribute(
        "n",
        session.getAttribute("n") == null ? 1 : (Integer)session.getAttribute("n") + 1
      );
    %>
    <h1><%= session.getAttribute("n") %></h1>
  </body>
</html>
```

Objetos complejos dentro de una sesión

Los valores que guardan los objetos de una sesión no tienen por qué ser siempre cadenas de caracteres o números, pueden ser cualquier tipo de dato, incluso la combinación de varios tipos.

En el siguiente ejemplo, vamos a cargar en una sesión tanto una lista de números enteros como un objeto de la clase `Persona`.

Para crear un objeto de la clase `Persona`, hay que definir primero esa clase. Como vimos en el Capítulo 12. Aplicaciones web en Java (JSP), en un proyecto jsp, las clases deben

ir en un paquete y no en la carpeta `web`. Por tanto, creamos el paquete `clases` y definimos `Persona` tal y como se muestra.

```
package clases;

public class Persona {
  public String nombre;
  public String cargo;

  public Persona(String nombre, String cargo) {
    this.nombre = nombre;
    this.cargo = cargo;
  }

  @Override
  public String toString() {
    return "Nombre: " + nombre + ", Cargo: " + cargo;
  }
}
```

En la página `index.jsp` se cargan los valores, tanto la lista de números como el objeto de la clase `Persona`.

```
<%@page import="clases.Persona"%>
<%@page import="java.util.ArrayList"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Objetos complejos en la sesión</title>
  </head>
  <body>
    <h1>Variables de sesión establecidas</h1>
    <%
      ArrayList<Integer> n = new ArrayList<Integer>();

      for (int i = 0; i < 10; i++) {
        n.add((int) (Math.random() * 10) + 1);
      }

      session.setAttribute("n", n);

      Persona alguien = new Persona("Susana Torio", "Programadora junior");
      session.setAttribute("alguien", alguien);
    %>
```

```
    <p>
      <a href="pagina1.jsp">Página 1</a>
    </p>
  </body>
</html>
```



Figura 15.4: Carga en la sesión una lista de números y un objeto de la clase `Persona`.

Y, por último, en `pagina1.jsp` se recuperan los valores. Observa cómo es necesario hacer *casting* para recuperar correctamente los objetos de la sesión.

```
<%@page import="java.util.ArrayList"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Objetos complejos en la sesión</title>
  </head>
  <body>
    <h1>Variables de sesión recuperadas</h1>
    <%
      for (int numero : (ArrayList<Integer>)session.getAttribute("n")) {
          out.print((Integer)numero + " ");
      }
    %>
    <hr>
    <%= session.getAttribute("alguien") %>
  </body>
</html>
```



Figura 15.5: Recupera los datos guardados en la sesión: una lista de números y un objeto de la clase `Persona`.

Login y control de usuarios

Las sesiones se crearon originalmente para guardar información de los usuarios que navegan por un sitio web. A día de hoy, las sesiones se siguen usando principalmente para ese cometido.

De forma típica, un usuario que entra en una aplicación web se *loguea* (inicia sesión) dando sus credenciales y una vez que se comprueba la validez de esas credenciales, se guardan en memoria ciertos datos que indican a la aplicación en todo momento quién es el usuario que está conectado.

Normalmente las aplicaciones web tienen una parte pública que puede ver cualquier usuario, sin necesidad de iniciar sesión, pero la parte más importante suele estar restringida a los usuarios *logueados*. Por ejemplo, en GMail[2], la parte pública es méramente informativa mientras que la parte realmente útil - la gestión del correo electrónico - está accesible después del inicio de sesión.

Realizaremos a continuación un control de usuarios básico pero más que suficiente para ilustrar este concepto.

Empezamos por la página principal, se trata del archivo `index.jsp`. Si el usuario no ha iniciado sesión, nos ofrecerá la opción de entrar (iniciar sesión). En caso de que el usuario se haya *logueado* previamente, la página mostrará el nombre de usuario.

[2] https://www.google.com/gmail/



Figura 15.6: Página principal con opción de inicio de sesión para un usuario.

El nombre de usuario se obtiene mediante `session.getAttribute("usuario")`. En caso de que el usuario no haya iniciado sesión o bien la haya cerrado, el valor devuelto será `null`. Por tanto es muy fácil saber si un usuario se ha *logueado* o no.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Página principal</title>
  </head>
  <body>
    <h1>Página principal</h1>
    <%
      if (session.getAttribute("usuario") == null) {
    %>
        <a href="formulario-login.jsp">Entrar</a>
    <%
      } else {
    %>
    Logueado como <a href="perfil.jsp"><%= session.getAttribute("usuario") %></a>
        (<a href="logout.jsp">Salir</a>)
    <%
    }
    %>

    <p>
      Página principal con información pública que puede ver cualquier usuario.
    </p>
  </body>
</html>
```

Al hacer clic en `Entrar`, la aplicación nos lleva a la página `formulario-login.jsp` donde se pide introducir un nombre de usuario y su correspondiente contraseña.

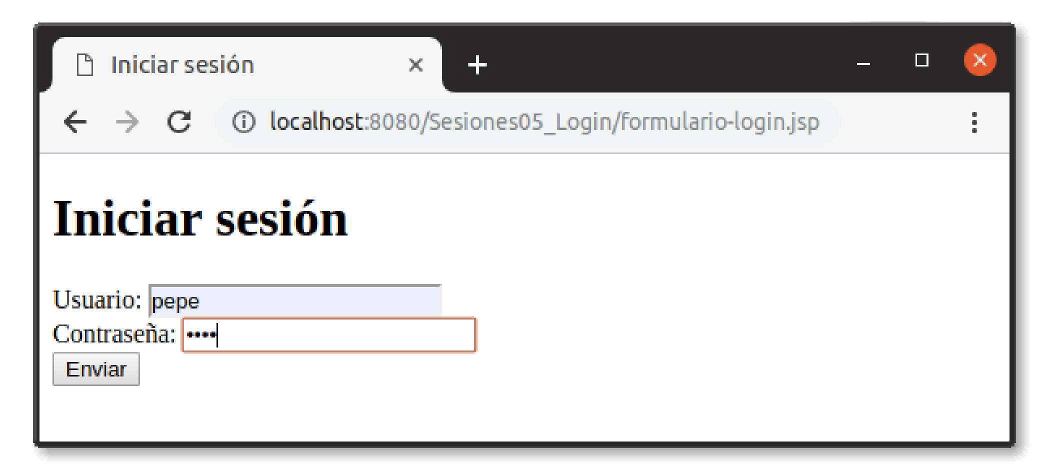


Figura 15.7: Formulario de inicio de sesión.

A continuación se muestra el código completo del formulario de inicio de sesión.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Iniciar sesión</title>
  </head>
  <body>
    <%
      if (session.getAttribute("usuario") != null) {
        response.sendRedirect("index.jsp");
      }
    %>
    <h1>Iniciar sesión</h1>

    <form action="login.jsp">
      Usuario: <input type="text" name="usuario"><br>
      Contraseña: <input type="password" name="contrasena"><br>
      <input type="submit" value="Enviar">
    </form>

    <div style="color: red;">
      <p>
      <%=
        session.getAttribute("error") == null ?
                "" : session.getAttribute("error")
      %>
```

```
      <% session.removeAttribute("error"); %>
      </p>
    </div>
  </body>
</html>
```

Observa que se establece un control al principio de la página por si alguien intenta acceder directamente habiendo iniciado sesión previamente. En tal caso, se redirige el flujo de ejecución a la página principal.

```
if (session.getAttribute("usuario") != null) {
  response.sendRedirect("index.jsp");
}
```

Justo debajo del formulario, hay una capa que muestra un mensaje de error en caso de producirse alguno. Una vez que se ha mostrado el mensaje de error es importante borrarlo con `session.removeAttribute("error")` para evitar que vuelva a aparecer al acceder de nuevo al formulario en otra ocasión.

```
<div style="color: red;">
  <p>
    <%=
      session.getAttribute("error") == null ?
              "" : session.getAttribute("error")
    %>
    <% session.removeAttribute("error"); %>
  </p>
</div>
```

Cuando se introduce el nombre de usuario y la contraseña y se le da al botón **Enviar**, la aplicación debe comprobar que los datos son correctos antes de dar por iniciada la sesión. Aunque el usuario que está usando la página no se percate, los datos se envian a la página `login.jsp` cuyo código se muestra a continuación.

```
<%
  String usuario = request.getParameter("usuario");
  String contrasena = request.getParameter("contrasena");

  if (usuario.equals("pepe") && contrasena.equals("1234")) {
    session.setAttribute("usuario", usuario);
    response.sendRedirect("index.jsp");
  } else {
    session.setAttribute("error", "Usuario o contraseña incorrecto");
    response.sendRedirect("formulario-login.jsp");
  }
%>
```

Para simplificar, la aplicación comprueba que el usuario es `pepe` y la contraseña es `1234`. Obviamente, en la práctica, esta comprobación debería ser algo más sofisticada. Los nombres de usuario y las contraseñas (convenientemente encriptadas) deberían estar guardadas en una base de datos y habría que hacer una consulta para comprobar la validez de la información introducida.

Lo que nos interesa ahora es comprobar si el nombre de usuario y la contraseña son correctos. En tal caso, guardamos en un objeto de la sesión el nombre del usuario mediante `session.setAttribute("usuario", usuario)` y la aplicación nos redirige a la página principal con `response.sendRedirect("index.jsp")`. Si los datos introducidos no son correctos, guardamos el error en la sesión mediante `session.setAttribute("error", "Usuario o contraseña incorrecto")` y volvemos otra vez al formulario de inicio de sesión con `response.sendRedirect("formulario-login.jsp")`.

Si el inicio de sesión se realiza con éxito, en la página principal, veremos el nombre de usuario, así como la opción **Salir** para cerrar la sesión.



Figura 15.8: Página principal con usuario logueado.

Al hacer clic sobre el nombre de usuario, la aplicación nos lleva a la página de perfil.



Figura 15.9: Página de perfil de usuario.

A continuación se muestra el código de `perfil.jsp`.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Perfil de usuario</title>
  </head>
  <body>
    <%
      if (session.getAttribute("usuario") == null) {
        session.setAttribute("error", "Debe iniciar sesión para acceder a la página de perfil.\
");
        response.sendRedirect("formulario-login.jsp");
      }
    %>
    <h1>Perfil de usuario</h1>

    <table>
      <tr>
        <td>
          <img src="user.png"></td>
        <td>
          Usuario: <%= session.getAttribute("usuario")%><br>
          Página de perfil con información del usuario.<br>
          <a href="index.jsp">Página principal</a>
        </td>
      </tr>
```

```
    </table>

  </body>
</html>
```

Un usuario malicioso podría intentar acceder a la página de perfil sin haber iniciado sesión. Para protegernos de esta situación, tenemos un trozo de código que comprueba si existe un usuario *logueado* y, en caso, negativo, nos redirige al formulario de inicio de sesión con el consiguiente mensaje de error.

```
if (session.getAttribute("usuario") == null) {
  session.setAttribute("error", "Debe iniciar sesión para acceder a la página de perfil.");
  response.sendRedirect("formulario-login.jsp");
}
```

En la siguiente captura podemos ver en color rojo el mensaje de error.



Figura 15.10: Mensaje de error en el formulario de inicio de sesión.

Otro posible error se produce cuando intentamos iniciar sesión con nombre de usuario y contraseña incorrectos.

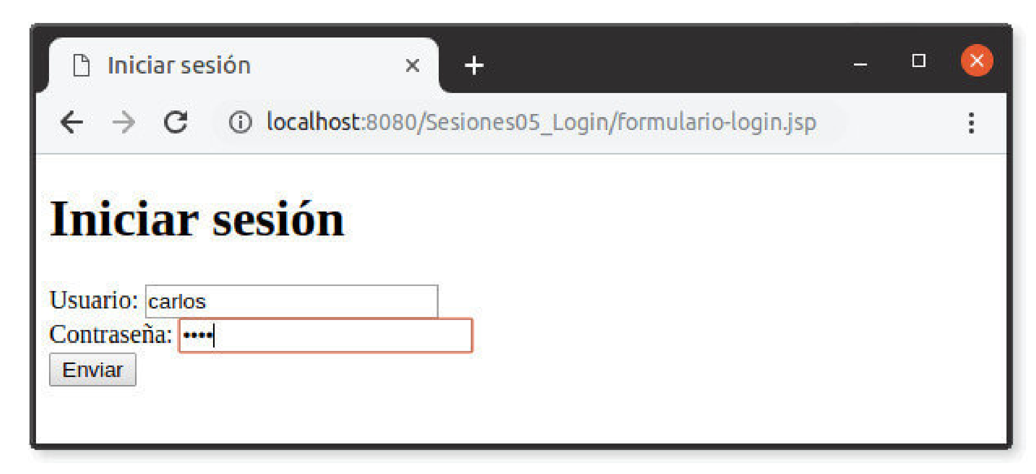


Figura 15.11: Formulario de inicio de sesión.

Al hacer clic en **Enviar**, el fichero `login.jsp` recibe los datos, comprueba que las credenciales no son correctas y, por tanto, establece el mensaje de error mediante `session.setAttribute("error", "Usuario o contraseña incorrecto")` y luego redirige la ejecución al formulario con `response.sendRedirect("formulario-login.jsp")` donde podemos ver en rojo el mensaje de error.



Figura 15.12: Mensaje de error en el formulario de inicio de sesión.

Encriptación de contraseñas

La información con nombres de usuario y contraseñas se suele guardar en una base de datos. Al tratarse de información sensible, es necesario tomar ciertas precauciones. Si alguien consiguiera acceder a esa base de datos, podría obtener un listado con las claves y estaría comprometiendo la seguridad de la aplicación. Por ello, es imprescindible en la práctica encriptar las contraseñas.

Lo que se guardará en la base de datos no será la contraseña sino un *hash* creado a partir de ella. Para una contraseña determinada, el *hash* siempre será el mismo.

Veamos un ejemplo de generación de un *hash* a partir de una contraseña que a fin de cuentas es una cadena de caracteres.

```
<%@page import="javax.xml.bind.DatatypeConverter"%>
<%@page import="java.security.MessageDigest"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSP Page</title>
  </head>
  <body>
    <h1>Encriptación de clave</h1>
    <%
      // Crea una instancia de la clase MessageDigest para manejar los
      // métodos de encriptación.
      MessageDigest md = MessageDigest.getInstance("MD5");
      // Actualiza la instancia con la cadena que queremos encriptar
      md.update("latarara".getBytes());
      // Crea el hash a partir de la cadena
      String hash = DatatypeConverter.printHexBinary(md.digest());

      // Muestra por pantalla el hash
      out.print("A la cadena \"latarara\" le corresponde el hash: " + hash + "<br>");

      // Comprueba si la clave es correcta
      out.print(
        MessageDigest.isEqual(hash.getBytes(), "26056A29E1E3A2E813EE5575A774B9B0".getBytes())
      );
    %>
  </body>
</html>
```

Se utiliza la clase `MessageDigest` para crear una instancia con la que manejar la encriptación.

```
MessageDigest md = MessageDigest.getInstance("MD5");
```

Observa que el método de encriptación que hemos utilizado es el MD5. La clase `MessageDigest` permite encriptar también usando los algoritmos SHA-1 y SHA-256.

Para poder encriptar, es necesario pasar la cadena de caracteres a bytes con el método `getBytes()`. Luego hay que actualizar la instancia con esos bytes.

```
md.update("latarara".getBytes());
```

En este ejemplo, la cadena que vamos a transformar en *hash* es “latarara”. Si el dato viene, por ejemplo, de un formulario habría que sustituir esa cadena por el dato recibido. Sería algo como `md.update(request.getParameter("contrasena").getBytes())`.

El método que realmente realiza la encriptación es `digest()`. Lo que se obtiene mediante este método son bytes; si queremos pasar esos bytes a una cadena de caracteres usaremos el método `DatatypeConverter.printHexBinary()`.

```
String hash = DatatypeConverter.printHexBinary(md.digest());
```

Para comprobar si dos cadenas *hash* son iguales, podemos utilizar el método `isEqual()` pasando previamente cada cadena a bytes.

```
out.print(
  MessageDigest.isEqual(hash.getBytes(), "26056A29E1E3A2E813EE5575A774B9B0".getBytes())
);
```

15.2 *Cookies*

Una *cookie* (galleta en inglés) es un fichero que se graba en el ordenador del propio usuario (no en el servidor) y que, por tanto, permite guardar información de manera local. Una *cookie* es algo parecido a una sesión aunque, a diferencia de esta última, la *cookie* no se borra cuando se cierra el navegador ni siquiera cuando se apaga el ordenador; para eliminarla hay que borrarla de forma explícita.

Las *cookies* se crean y se gestionan mediante la clase homónima: `Cookie`.

En realidad, las sesiones están implementadas como *cookies* internamente. Cada vez que se inicia una sesión, se crea una *cookie* llamada `JSSESIONID` que expira cuando se cierra la sesión.

Creación de *cookies*

Para crear una *cookie* se usa el constructor `Cookie()`.

```
Cookie(String name, String value)
```

A cada *cookie* se le da un nombre y un valor, igual que cuando se asigna un objeto a una sesión. A diferencia de las sesiones en las que podíamos asignar objetos de cualquier tipo como por ejemplo números enteros, instancias o listas; en una *cookie* solo podemos guardar una cadena de caracteres. A pesar de esta limitación, existen algunos "trucos" para almacenar objetos complejos en cadenas de caracteres como veremos más adelante.

Por ejemplo, podemos dar a `numero` el valor `24`.

```
new Cookie("numero", "24");
```

Todavía no tenemos la *cookie* guardada correctamente en el disco duro. Hace falta añadir el objeto creado a `response` que es la respuesta que el servidor le da al cliente. En román paladino, el servidor le tiene que decir al cliente: "Oye navegador, guarda esta cookie en un fichero". Veamos el código completo.

```
Cookie numero = new Cookie("numero", "24");
Cookie color = new Cookie("color", "rojo");
Cookie secuencia = new Cookie("secuencia", "55 21 48 97 33");
response.addCookie(numero);
response.addCookie(color);
response.addCookie(secuencia);
```

Si ejecutamos este archivo JSP[3] no veremos nada en el navegador porque no hay ninguna instrucción para volcar contenido en el HTML.

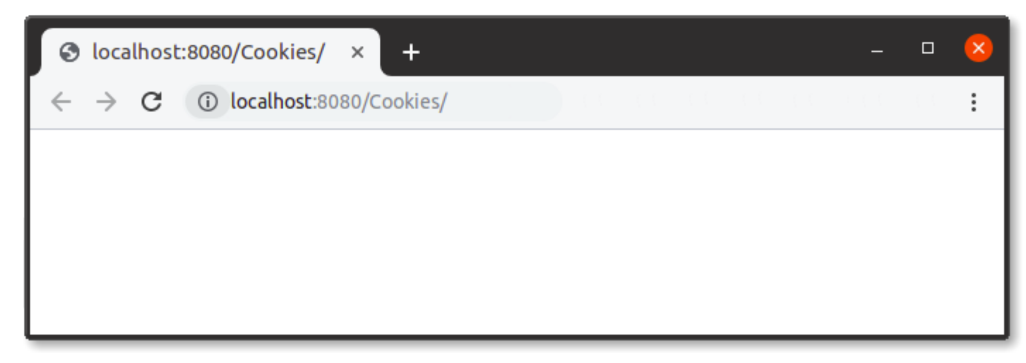


Figura 15.13: Creación de *cookies*.

¿Qué ha sucedido entonces? Se han grabado en el disco duro de nuestro ordenador, en un fichero, los datos referentes a las *cookies* que hemos indicado.

Dependiendo del navegador utilizado y de la versión, estos datos se pueden ver de diferente manera. Veamos cómo se pueden ver las *cookies* creadas en **Chrome**.

Despliega el menú (tres puntos) y selecciona **Configuración**.

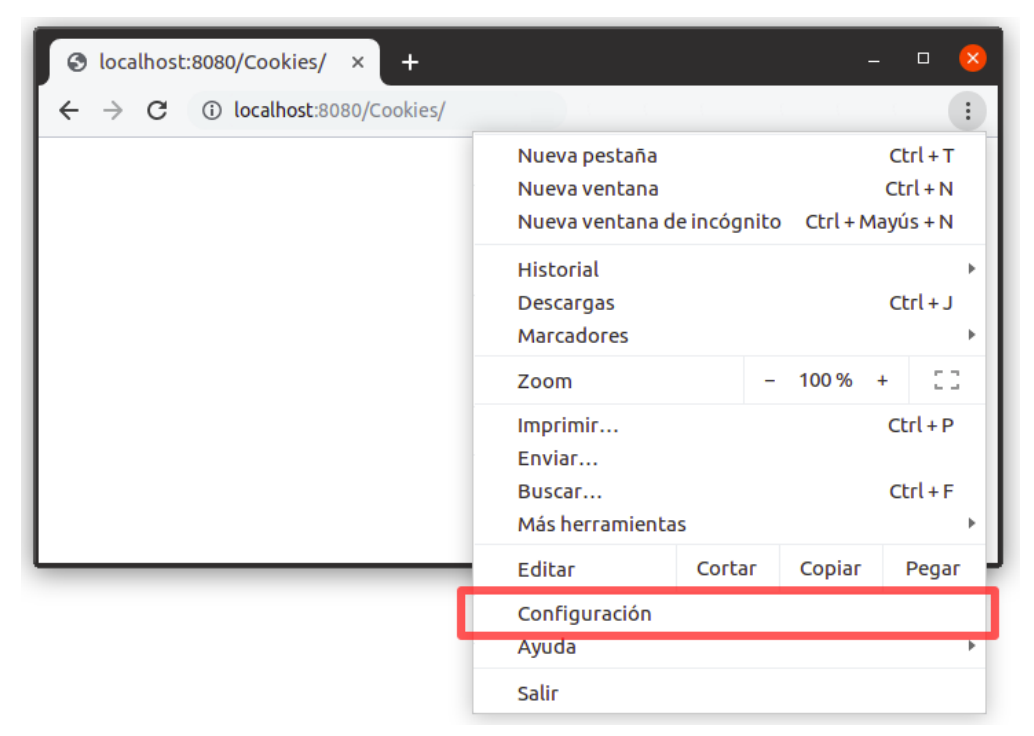


Figura 15.14: Configuración.

Desplázate por las opciones hacia abajo y selecciona **Configuración avanzada**.

[3] https://raw.githubusercontent.com/LuisJoseSanchez/aprende-java-con-ejercicios/master/ejemplos/15_Sesiones_y_cookies/Cookies01_Creacion_de_cookies/index.jsp

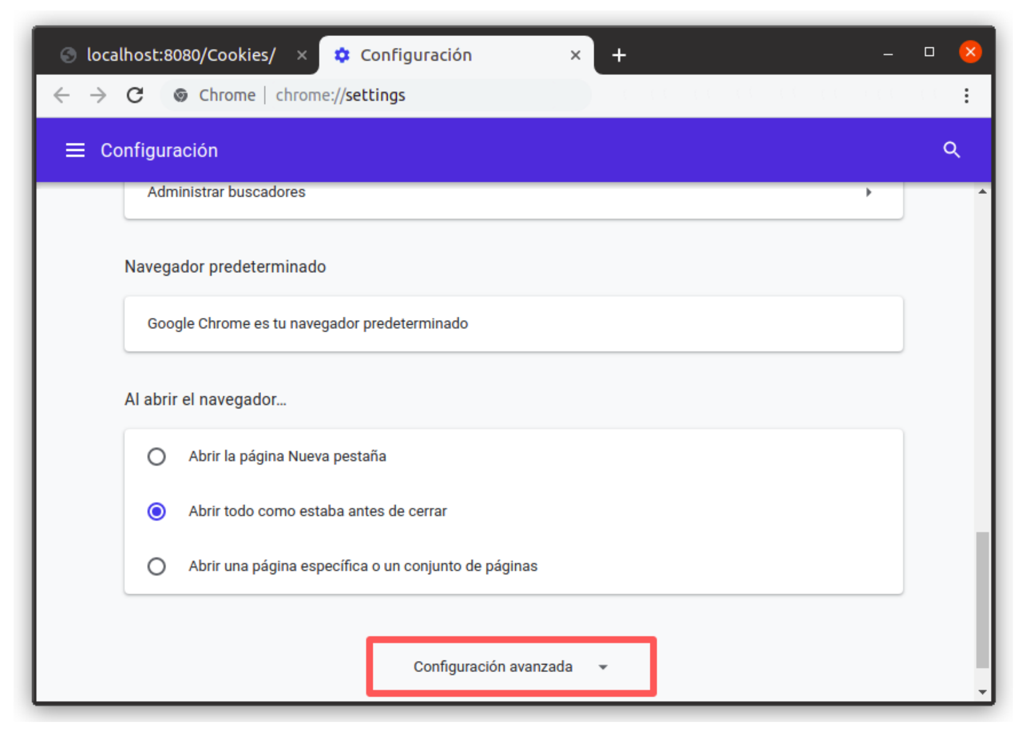


Figura 15.15: Configuración avanzada.

Busca la opción **Configuración del sitio web**.

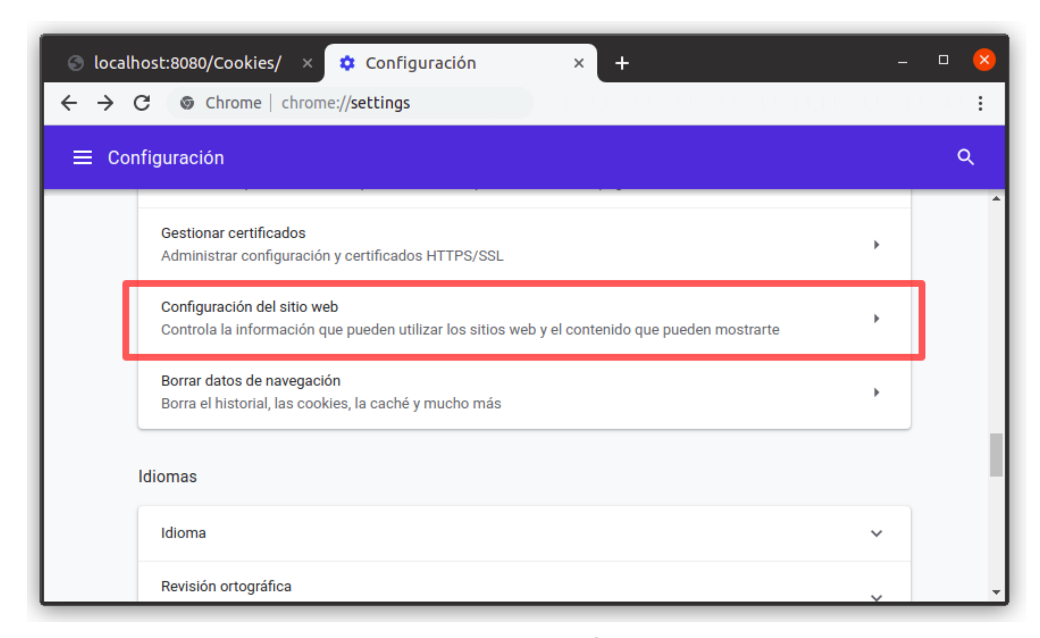


Figura 15.16: Configuración del sitio web.

Selecciona **Cookies**.

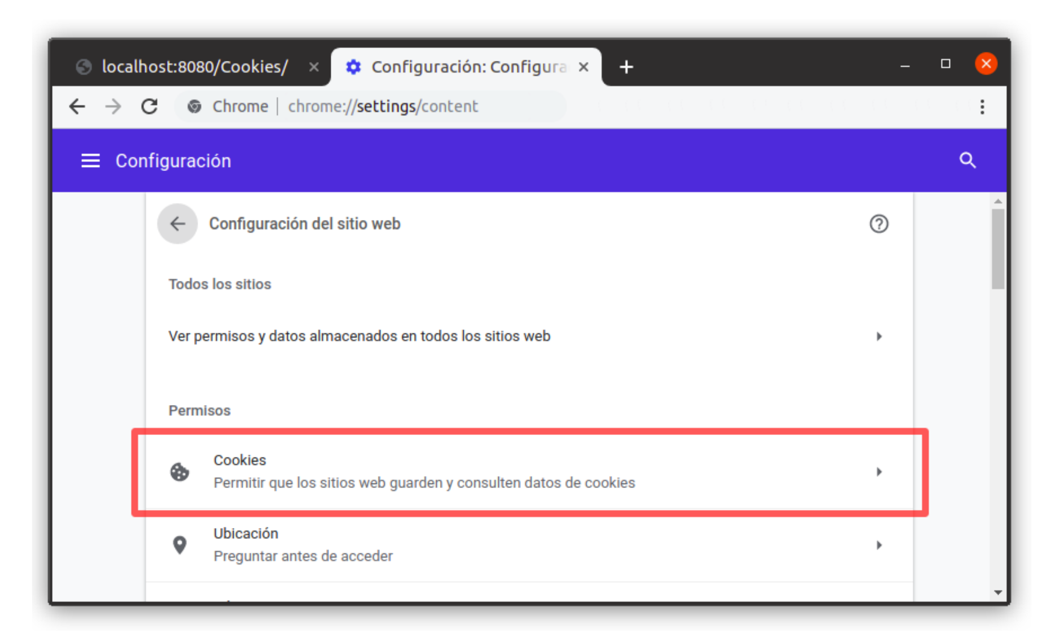


Figura 15.17: Cookies.

Selecciona **Ver todas las cookies y datos del sitio web**.

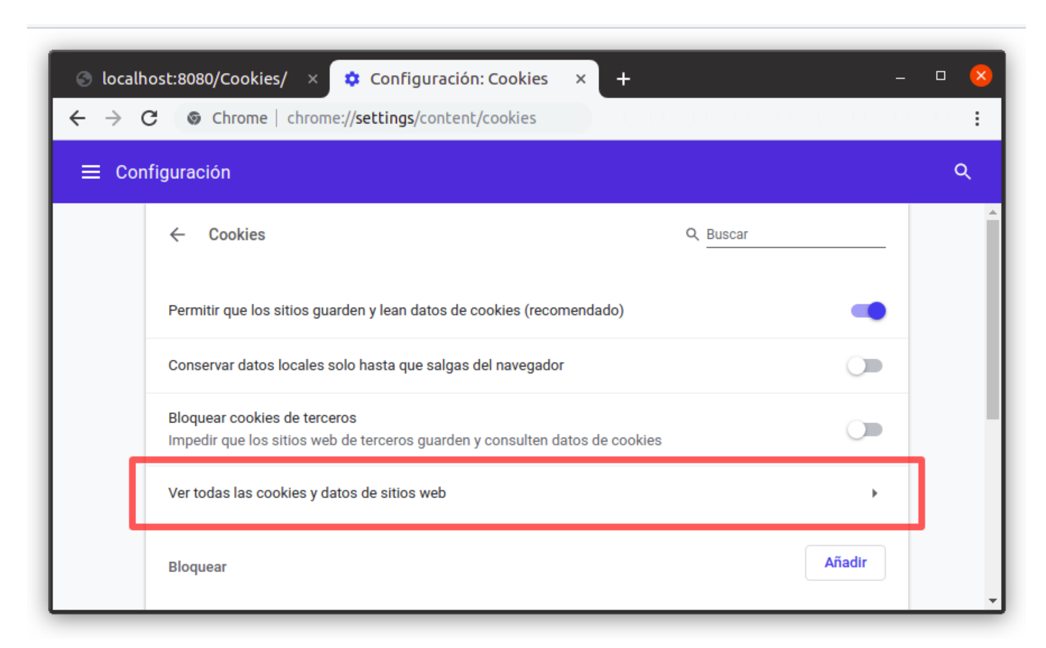


Figura 15.18: Ver todas las cookies y datos del sitio web.

Seguramente tendrás muchas **cookies** de diferentes sitios web. Las que hemos creado anteriormente con nuestro ejemplo están en **localhost**. En el cuadro de búsqueda, escribe **local**. Verás cómo se filtra el listado de modo que solo aparece **localhost**; selecciona esta opción.

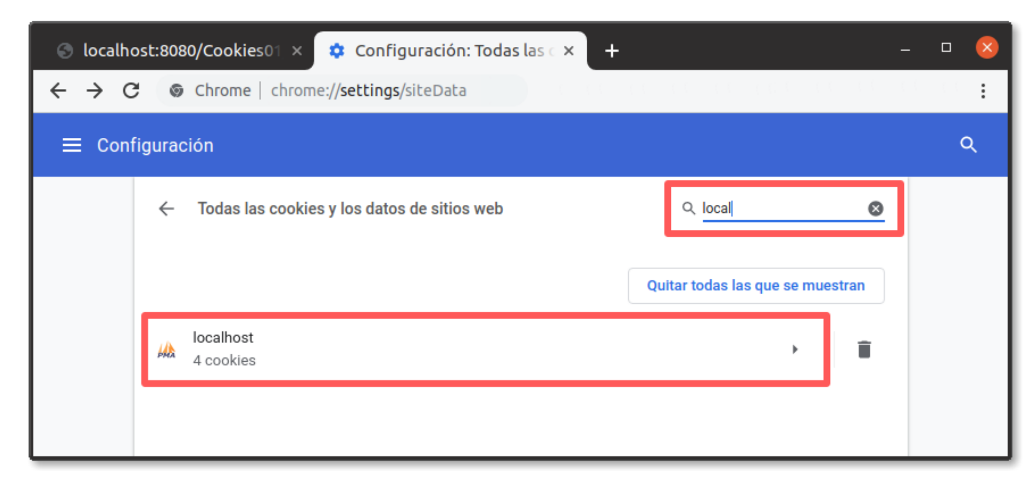


Figura 15.19: Búsqueda .

Por fin hemos accedido a nuestras *cookies* **color**, **número** y **secuencia**. Veamos cualquiera de ellas, por ejemplo “color”.

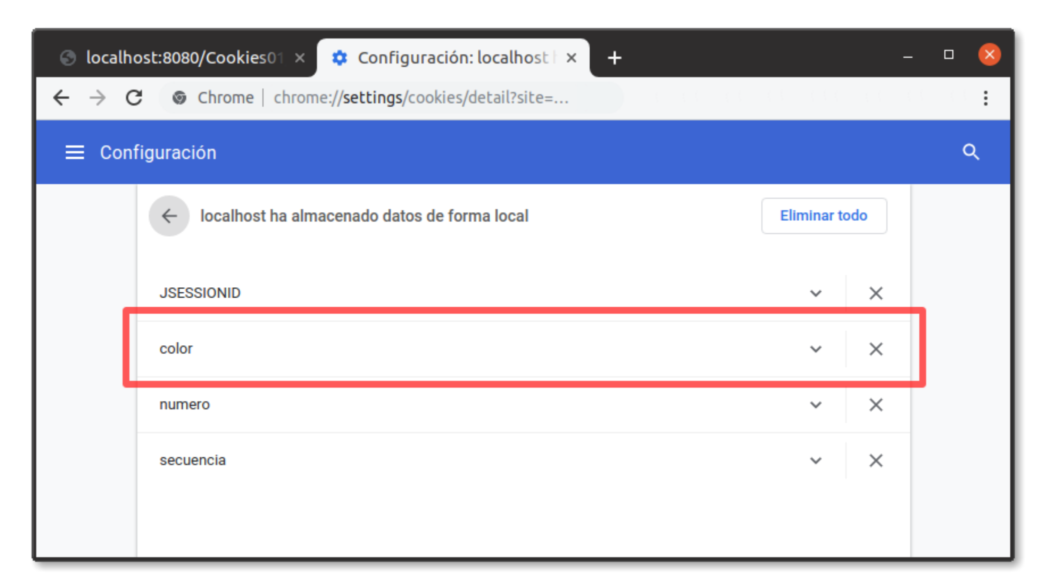


Figura 15.20: Cookies de ejemplo: color, número y secuencia.

Volià, efectivamente el valor de “color” es “rojo”.

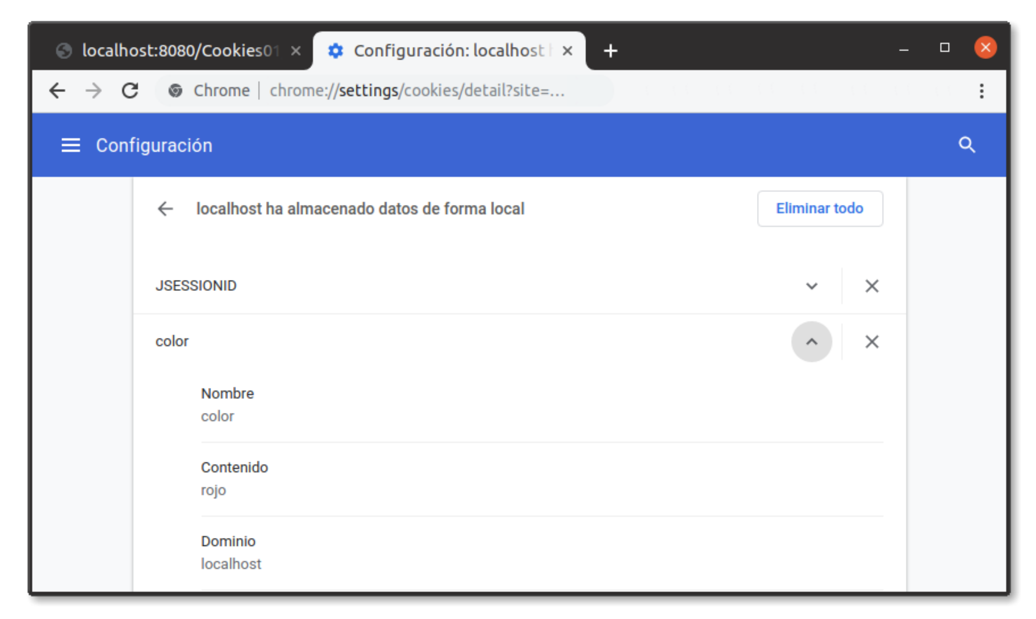


Figura 15.21: Cookie color.

Ruta y caducidad de una *cookie*

Cada *cookie* se graba en una determinada ruta; por defecto en Netbeans, cuando no se especifica ninguna, la ruta de la *cookie* coincide con el nombre del proyecto. Por ejemplo, si el proyecto se llama **Cookies01**, la ruta será **/Cookies01**.

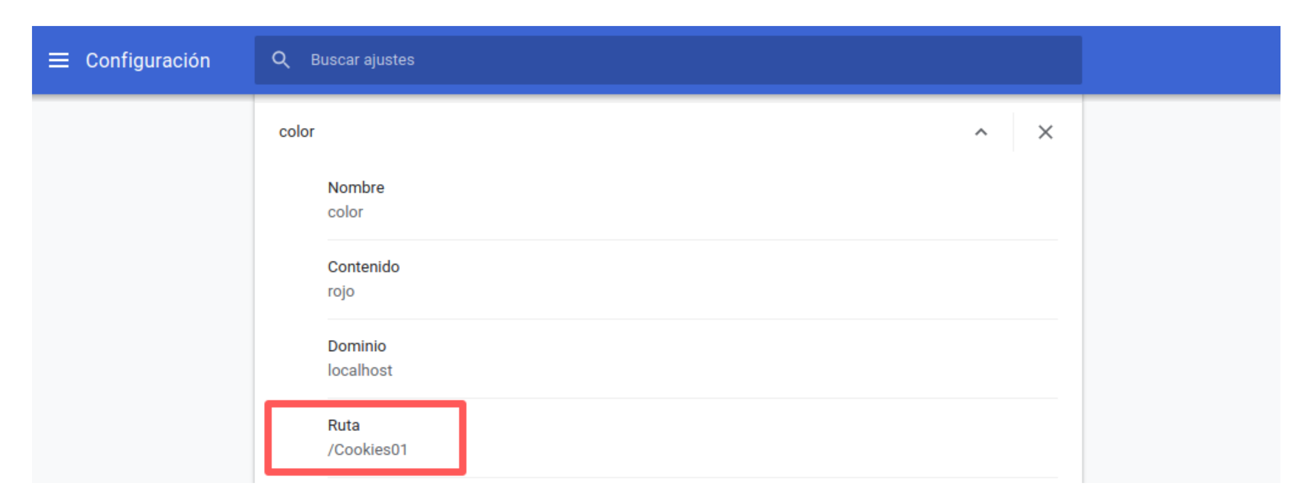


Figura 15.22: Ruta de la cookie color.

La ruta de una *cookie* se puede cambiar con el método `setPath()`.

También se puede especificar la caducidad, es decir, el tiempo de vida. Esto se hace mediante el método `setMaxAge()` que toma como parámetro el tiempo en segundos que durará la *cookie*, siempre que ese parámetro sea un número positivo. Si se proporciona un número negativo, la *cookie* se borrará cuando se cierre el navegador.

Vamos a ampliar el ejemplo anterior especificando una ruta y una caducidad para cada *cookie*.

```
int semana = 60 * 60 * 24 * 7;

Cookie numero = new Cookie("numero", "24");
numero.setPath("/");
numero.setMaxAge(semana);
response.addCookie(numero);

Cookie color = new Cookie("color", "rojo");
color.setPath("/");
color.setMaxAge(semana);
response.addCookie(color);

Cookie secuencia = new Cookie("secuencia", "55 21 48 97 33");
secuencia.setPath("/");
secuencia.setMaxAge(semana);
response.addCookie(secuencia);
```

Cómo obtener todas las *cookies*

Veamos con un programa de ejemplo cómo podemos obtener todos los valores de las cookies almacenadas en `localhost`. La clave está en que `request.getCookies()` devuelve un array de *cookies* que podemos recorrer con un simple bucle *for*.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Cookies</title>
  </head>
  <body>
    <%
      Cookie[] cookies = request.getCookies();

      if (cookies != null) {
        for (Cookie cookie : cookies) {
          out.println(cookie.getName() + ": " + cookie.getValue() + "<br>");
        }
      }
    %>
  </body>
</html>
```

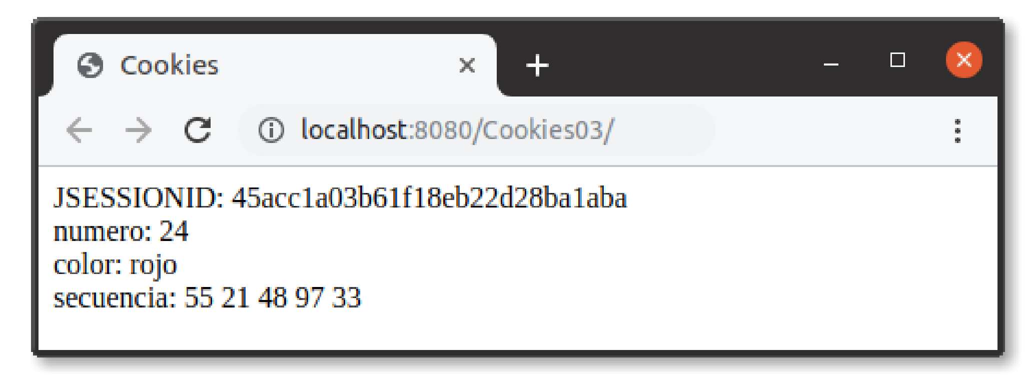


Figura 15.23: Muestra todas las cookies de localhost.

Cómo obtener una *cookie* concreta

Para obtener el valor de una *cookie* que tiene un nombre concreto hay que recorrer el array de cookies visto en el ejemplo anterior. Si es algo que se va a realizar con frecuencia, lo mejor es aislar el código necesario en una pequeña función que se llame por ejemplo `dameCookie()` y llamarla cada vez que haga falta.

Lo veremos muy claro con un ejemplo. En primer lugar se le pide al usuario el nombre de una *cookie* mediante un formulario.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Obtiene el valor de una cookie</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
  </head>
  <body>
    <form action="cookie.jsp" method="POST">
      Nombre de la <i>cookie</i> que quiere consultar
      <input type="text" name="nombre"> <br>
      <input type="submit" value="Aceptar">
    </form>
  </body>
</html>
```

El nombre se envía a `cookie.jsp`. La función `dameCookie()` devuelve la *cookie* correspondiente si existe, o el valor `null` en caso de que no exista.

```
<%@page import="org.jboss.weld.context.http.HttpRequestContext"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Obtiene el valor de una cookie</title>
  </head>
  <body>
    <%
      String nombre = request.getParameter("nombre");

      Cookie cookie = dameCookie(request, nombre);

      if (cookie == null) {
        out.println("No se ha encontrado ninguna cookie con ese nombre.");
      } else {
        out.println("El valor de la cookie " + nombre + " es " + cookie.getValue());
      }
    %>
  </body>
</html>

<%!
  public static Cookie dameCookie(HttpServletRequest request, String nombre) {
    Cookie[] cookies = request.getCookies();

    if (cookies != null) {
      for (Cookie cookie : cookies) {
        if (cookie.getName().equals(nombre)) {
          return cookie;
        }
      }
    }
    return null;
  }
%>
```

Objetos complejos dentro de una cookie

Hemos visto que una *cookie* se graba siembre como una cadena de texto ¿qué podemos hacer si queremos guardar objetos de una determinada clase, un array, una lista o cualquier otra estructura que no sea una simple cadena? Afortunadamente hay una solución, una estructura compleja se puede “aplanar” o convertir en un texto con

formato JSON. Lógicamente, para recuperar la estructura tendremos que realizar justo lo contrario, es decir, pasar el JSON a la estructura correspondiente.

Es conveniente recordar que el tamaño máximo de una *cookie* es de 4Mb por lo que, si vamos a guardar mucha información en local, quizás debamos plantearnos usar el almacenamiento que proporciona HTML5.

Para manipular datos en formato JSON vamos a usar la librería Jackson[4]. Necesitaremos descargar los tres archivos jar necesarios que están en https://github.com/LuisJoseSanchez/aprende-java-con-ejercicios/tree/master/ejemplos/15_Sesiones_y_cookies/Cookies05_Objeto_en_una_cookie/jackson_jar_files y añadirlos a las librerías del proyecto. Para ello, haz clic con el botón derecho en **Libraries** y selecciona **Agregar archivo JAR/Carpeta**.

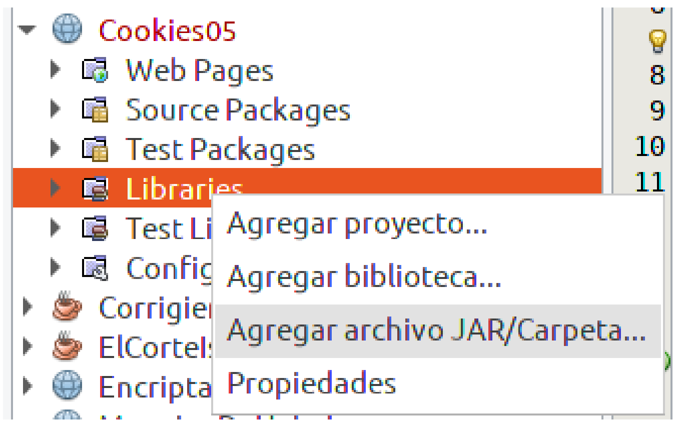


Figura 15.24: Añadiendo archivos jar.

Añade los tres archivos jar.

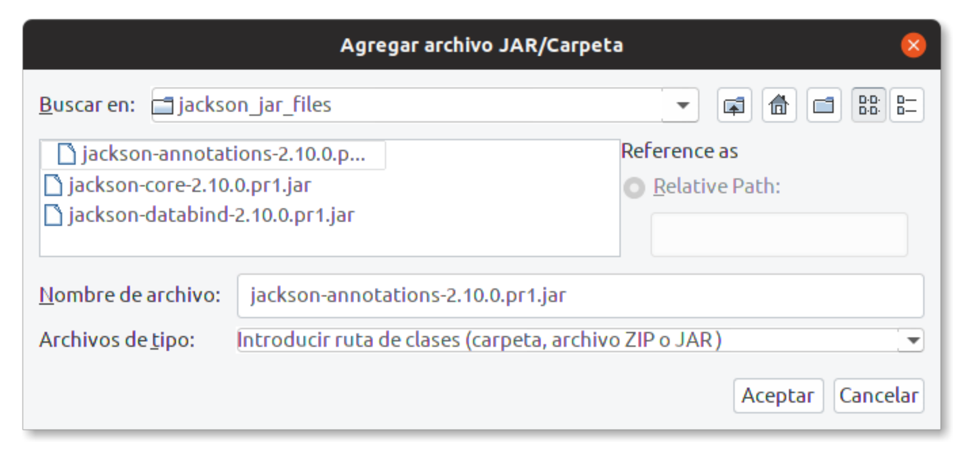


Figura 15.25: Añadiendo archivos jar.

Una vez añadida la librería Jackson, ya podemos convertir datos a formato JSON.

En este ejemplo vamos a guardar un objeto de la clase `Persona` en una cookie. A

[4] Página oficial del proyecto Jackson en GitHub: https://github.com/FasterXML/jackson

continuación se muestra el código de esta clase, ten en cuenta que se ha creado dentro del paquete `clases`. Para que funcione bien el ejemplo, hemos tenido que definir un constructor vacío para `Persona`.

```
package clases;

public class Persona implements java.io.Serializable {
  public String nombre;
  public String cargo;

  public Persona() {
  }

  public Persona(String nombre, String cargo) {
    this.nombre = nombre;
    this.cargo = cargo;
  }

  @Override
  public String toString() {
    return "Nombre: " + nombre + ", Cargo: " + cargo;
  }
}
```

En `index.jsp` se crea el objeto de la clase `Persona`, se convierte en un texto en formato JSON y se guarda en una cookie llamada `alguien`.

```
<%@page import="com.fasterxml.jackson.databind.ObjectMapper"%>
<%@page import="clases.Persona"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Objeto en una cookie</title>
  </head>
  <body>
    <%
      Persona alguien = new Persona("Susana Torio", "Programadora junior");
      ObjectMapper mapper = new ObjectMapper();
      String json = mapper.writeValueAsString(alguien);
      Cookie cookie = new Cookie("alguien", json);
      cookie.setPath("/");
      response.addCookie(cookie);
      out.println("Se ha grabado <b>" + json + "</b> en la cookie <b>alguien</b>");
    %>
```

```
    <p>
      <a href="cookie.jsp">Recuperar la cookie desde otra página</a>
    </p>
  </body>
</html>
```



Figura 15.26: Grabación de un objeto en una cookie.

Haciendo clic en el enlace **Recuperar la cookie desde otra página** nos vamos a `cookie.jsp` que se encarga de recuperar la cookie guardada. Primero se recupera el JSON y luego se transforma en un objeto de la clase `Persona`.

```
<%@page import="com.fasterxml.jackson.databind.ObjectMapper"%>
<%@page import="clases.Persona"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Objeto en una cookie</title>
  </head>
  <body>
    <%
      Cookie cookie = dameCookie(request, "alguien");
      ObjectMapper objectMapper = new ObjectMapper();
      Persona p = objectMapper.readValue(cookie.getValue(), Persona.class);
      out.println("Contenido de la cookie <b>alguien</b>:<br>");
      out.println(p);
    %>
  </body>
</html>

<%!
  public static Cookie dameCookie(HttpServletRequest request, String nombre) {
    Cookie[] cookies = request.getCookies();

    if (cookies != null) {
      for (Cookie cookie : cookies) {
```

```
        if (cookie.getName().equals(nombre)) {
          return cookie;
        }
      }
    }
    return null;
  }
%>
```

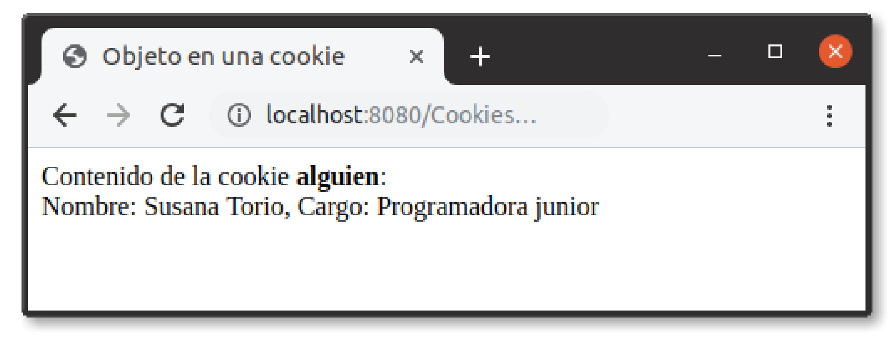


Figura 15.27: Recuperación de un objeto guardado en una cookie.

15.3 Ejercicios

Sesiones



Ejercicio 1

Modifica el contador de visitas visto como ejemplo en este capítulo añadiendo la posibilidad (mediante un enlace o botón) de reiniciar la cuenta. Utiliza el método `invalidate()` para resetear las variables de sesión.



Ejercicio 2

Escribe un programa que calcule la media de un conjunto de números positivos introducidos por el usuario mediante un formulario (se introduce un número por cada carga de página). A priori, el programa no sabe cuántos números se introducirán. El usuario indicará que ha terminado de introducir los datos cuando meta un número negativo (este último número no se cuenta en la media).



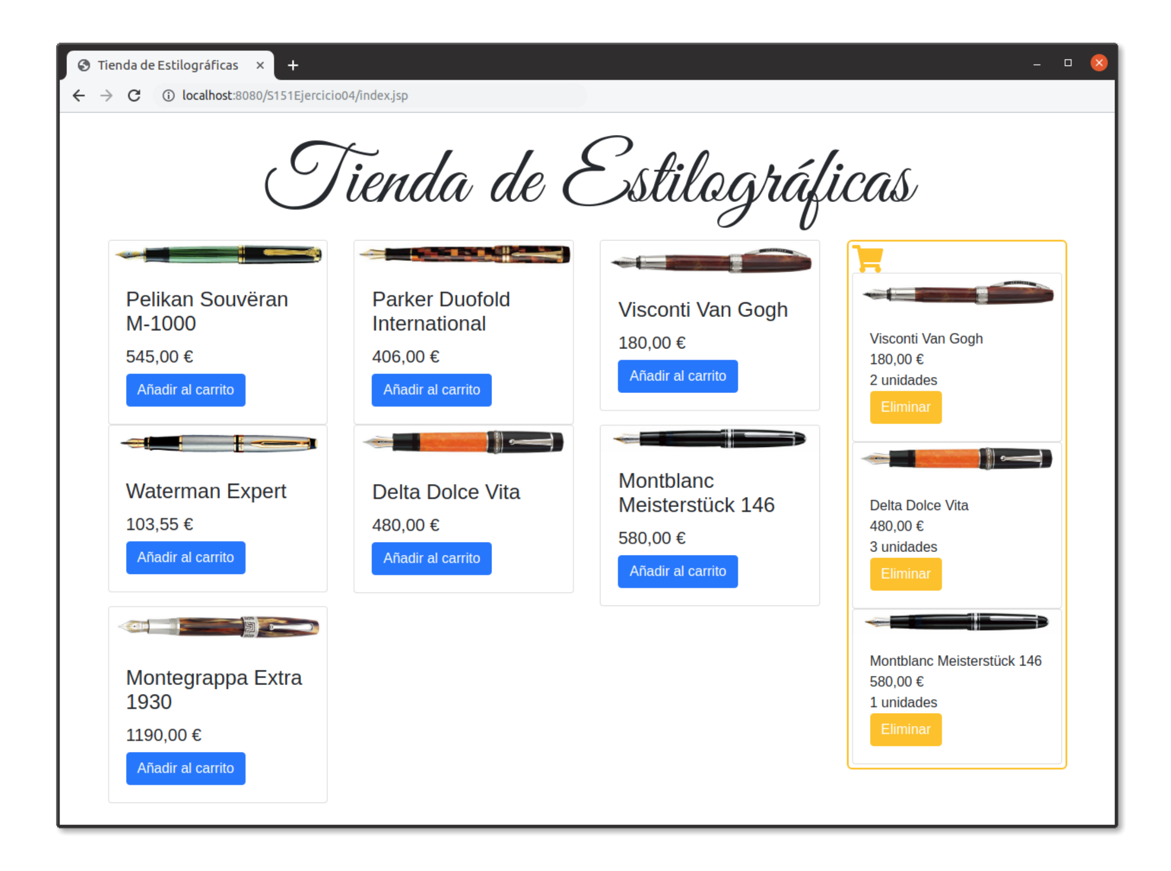
Ejercicio 3

Realiza un programa que vaya pidiendo números hasta que se introduzca un numero negativo y nos diga cuantos números se han introducido, la media de los impares y el mayor de los pares. El número negativo sólo se utiliza para indicar el final de la introducción de datos pero no se incluye en el cómputo.



Ejercicio 4

Crea una tienda *on-line* sencilla con un catálogo de productos y un carrito de la compra. Un catálogo de cuatro o cinco productos será suficiente. De cada producto se debe conocer al menos el nombre, el precio y una imagen. Todos los productos deben tener una imagen que los identifique. Al lado de cada producto del catálogo deberá aparecer un botón **Comprar** que permita añadirlo al carrito. Si el usuario hace clic en el botón **Comprar** de un producto que ya estaba en el carrito, se deberá incrementar el número de unidades de dicho producto. Para cada producto que aparece en el carrito, habrá un botón **Eliminar** por si el usuario se arrepiente y quiere quitar un producto concreto del carrito de la compra. A continuación se muestra una captura de pantalla de una posible solución.


Tienda de Estilográficas
localhost:8080/S151Ejercicio04/index.jsp
Tienda de Estilográficas
Pelikan Souvëran M-1000
545,00 €
Añadir al carrito
Parker Duofold International
406,00 €
Añadir al carrito
Visconti Van Gogh
180,00 €
Añadir al carrito
Waterman Expert
103,55 €
Añadir al carrito
Delta Dolce Vita
480,00 €
Añadir al carrito
Montblanc Meisterstück 146
580,00 €
Añadir al carrito
Montegrappa Extra 1930
1190,00 €
Añadir al carrito
Visconti Van Gogh
180,00 €
2 unidades
Eliminar
Delta Dolce Vita
480,00 €
3 unidades
Eliminar
Montblanc Meisterstück 146
580,00 €
1 unidades
Eliminar

Cookies



Ejercicio 1

Realiza una aplicación que permita elegir el idioma y guarde la opción elegida en una *cookie*. Simplemente se mostrará el título de la página y un párrafo en el idioma seleccionado. Cuando el usuario cierre el navegador y posteriormente lo vuelva a abrir, debería estar en el idioma que se eligió la última vez.





Ejercicio 2

Escribe un programa que guarde en una *cookie* el color de fondo (propiedad `background-color`) de una página y el color de la fuente (propiedad `color`). Esta página debe tener únicamente algo de texto y un formulario para cambiar los colores. Cuando el usuario cierre el navegador y posteriormente lo vuelva a abrir, deberían aparecer los colores que se eligieron la última vez.



Apéndice A. Ejercicios de ampliación



Ejercicio 1: Transcomunicación instrumental mediante ficheros de texto digitales

Se conoce la **transcomunicación instrumental** (TCI) como la técnica de comunicación que utiliza aparatos electrónicos para establecer comunicación con los espíritus. El caso de TCI más conocido es el de la psicofonía.

Para realizar una psicofonía se pone a grabar al aire un dispositivo en un lugar donde previamente hayan ocurrido fenómenos extraños o simplemente se invoca la entidad que queremos que se comunique mediante el ritual adecuado. Posteriormente, se escucha la grabación para comprobar si se ha grabado algo: un mensaje, una palabra, un grito, un lamento…

Los sonidos extraños se graban gracias a lo que se llama en términos parapsicológicos una señal portadora, puede ser el chasquido de una rama, una racha de viento, un golpe. La entidad comunicante aprovecha ese sonido para supuestamente moldearlo y convertirlo en un mensaje inteligible. Muchas de las psicofonías se realizan utilizando como portadora el ruido que produce una radio que no está sintonizada.

El ejercicio que se propone es la realización de un **programa que permita realizar la TCI mediante la creación y el posterior análisis de ficheros de texto**.

En primer lugar, el programa deberá generar la portadora, que será un fichero de texto creado con caracteres producidos de forma aleatoria. Se podrán especificar varios parámetros como el número de líneas o páginas que se quieren generar, si se incluyen letras únicamente o bien letras y números, frecuencia aproximada de espacios, saltos de línea y signos de puntuación, etc. Se obtienen portadoras de calidad - con más probabilidad de contener mensajes - cuando la proporción de las letras que contienen éstas coincide con las del idioma español.

Frecuencia de aparición de letras:

http://es.wikipedia.org/wiki/Frecuencia_de_aparici%C3%B3n_de_letras

Una vez creado el fichero de texto, el programa deberá ser capaz de analizarlo, al menos de una forma superficial, de modo que nos resulte fácil encontrar los posibles mensajes. Para realizar este análisis, el programa irá comparando una a una todas las palabras que aparecen en el texto con las que hay en el diccionario de español. En caso de coincidencia, la palabra o palabras encontradas deben aparecer resaltadas con un color diferente al resto del texto o bien en vídeo inverso. Opcionalmente se pueden mostrar estadísticas con el número total de palabras encontradas, distancia media entre esas palabras, palabras reales por cada mil palabras generadas, etc.

El alumno puede encontrar ficheros con las palabras que tiene el idioma español en formato digital en la siguiente dirección. Se trata de una muestra, hay otros muchos diccionarios disponibles en internet:

http://lasr.cs.ucla.edu/geoff/ispell-dictionaries.html#Spanish-dicts



Ejercicio 2: Colección de discos ampliado (con canciones)

Este ejercicio consiste en realizar una mejora del ejercicio del capítulo 10 que permite gestionar una colección de discos. Ahora cada disco contiene canciones. Deberás crear la clase `Cancion` con los atributos y métodos que estimes oportunos. Añade el atributo `canciones` a la clase `Disco`. Este atributo `canciones` debe ser un array de objetos de la clase `Cancion`.

Fíjate bien que `Cancion` NO ES una subclase de `Disco` sino que ahora cada disco puede tener muchas canciones, que están almacenadas en el atributo `canciones`.

Modifica convenientemente el método `toString` para que al mostrarse los datos de un disco, se muestre también la información sobre las canciones que contiene.

La aplicación debe permitir añadir, borrar y modificar las canciones de los discos.

Apéndice B. Entorno de Desarrollo Integrado Netbeans

Netbeans es un IDE (Integrated Development Environment), es decir, un **Entorno de Desarrollo Integrado**. Consta de un potente editor y de todas las herramientas necesarias para codificar, compilar, depurar y ejecutar una aplicación.

Este IDE tiene algunas características que lo hacen muy atractivo para los programadores:

- Es el IDE oficial para desarrollar en Java[5].
- Chequea errores de sintaxis y da advertencias al mismo tiempo que se escribe el código.
- Tiene autocompletado inteligente de palabras clave, clases, métodos, parámetros…
- Permite depurar una aplicación ejecutándola paso a paso y ver el contendido de todas las variables en tiempo real.
- Soporta muchos lenguajes de programación además de Java: PHP, Groovy, HTML, XML, Javascript, JSP…
- Existe una gran variedad de *plugins* que amplían las funcionalidades del entorno.

Descarga e instalación.

Netbeans se puede descargar de forma gratuita y para cualquier plataforma desde https://netbeans.org/downloads/.

[5] Esto no es casualidad ya que es la misma empresa - **Oracle** - la que está detrás del desarrollo de Java y también la que soporta **Netbeans**.

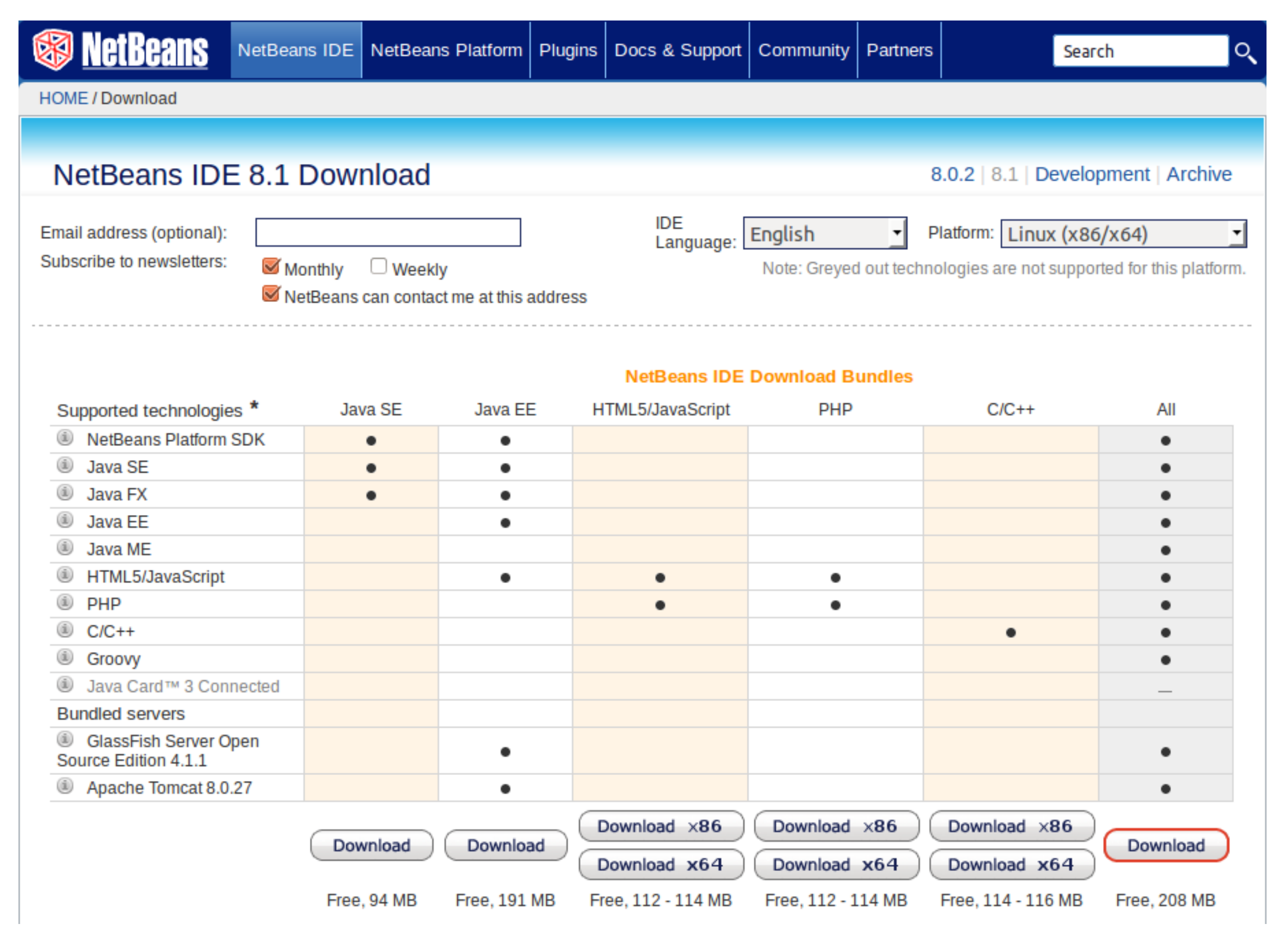


Figura B.1: Descarga de Netbeans

El programa de instalación de **Netbeans** está disponible en varios paquetes según el lenguaje de programación y las tecnologías que se incluyen. En el momento de escribir este libro, los paquetes disponibles son: Java SE, Java EE, HTML5/JavaScript, PHP, C/C++, All. Este último paquete (All) instala todos los lenguajes y tecnologías disponibles para el IDE (sin contar los *plugins*). Para seguir este libro es suficiente con la versión Java EE, sin embargo animo al lector a descargar la versión completa y experimentar con otros lenguajes diferentes a Java, por ejemplo con PHP[6].

Configuración

En los ejemplos y soluciones a los ejercicios de este libro nos ceñimos al los estándares de programación de Google para el código fuente escrito en el lenguaje de programación Java[7]. Por tanto, después de la instalación de **Netbeans**, es necesario realizar algunso ajustes en la configuración del editor para cumplir con las siguientes

[6] https://leanpub.com/aprendephpconejercicios/

[7] https://google.github.io/styleguide/javaguide.html

condiciones:

- La tecla TAB debe insertar espacios, no debe insertar el carácter de tabulación.
- La indentación debe ser de 2 caracteres (por defecto suele estar a 4).
- La codificación de caracteres debe ser UTF-8.

Para ello selecciona **Herramientas** → **Opciones** → **Editor**. A continuación selecciona la pestaña **Formato** y en el menú desplegable correspondiente al lenguaje de programación selecciona **Java**. La configuración debería quedar como se indica en la imagen.

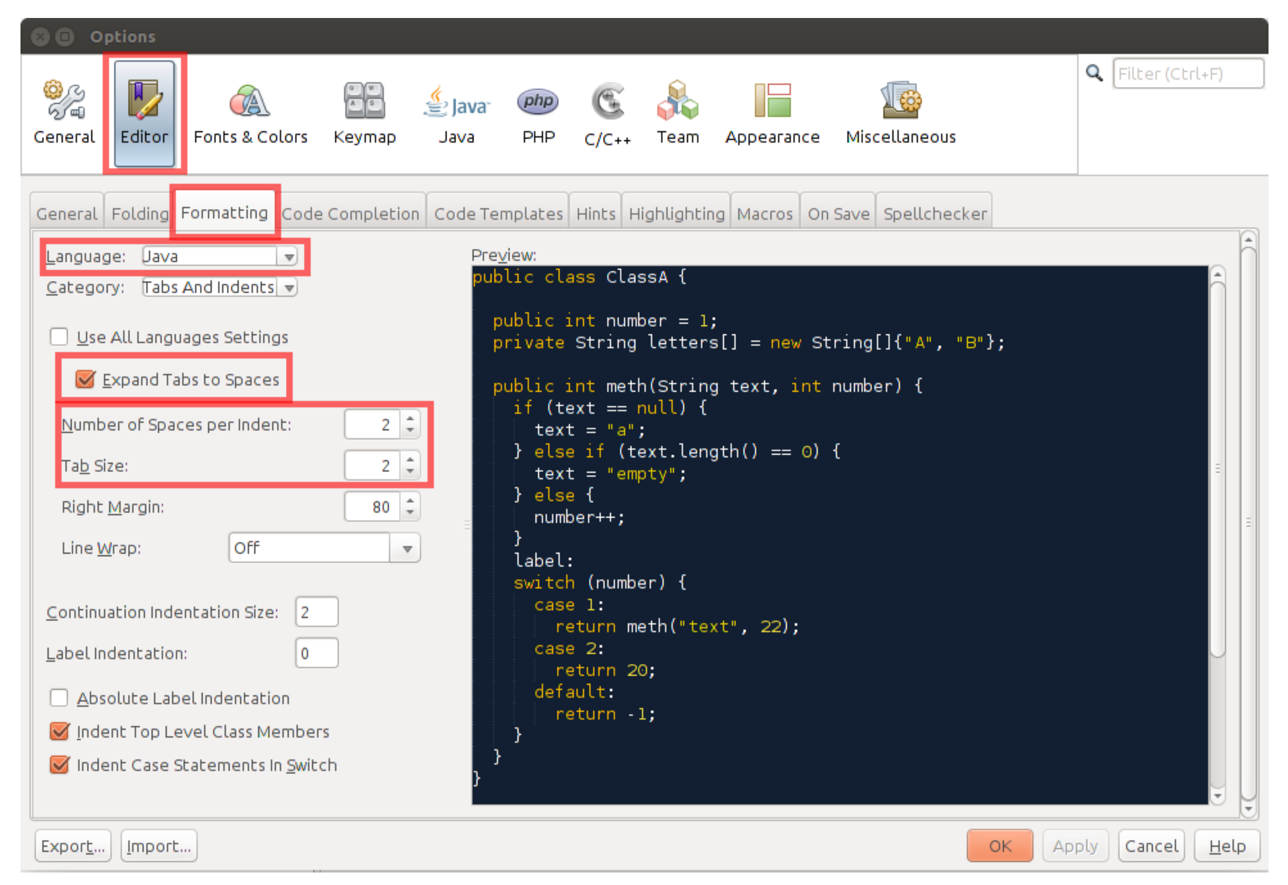


Figura B.2: Configuración de Netbeans

Creación de un proyecto

Sigue los pasos que damos a continuación para crear un nuevo proyecto de Java en **Netbeans**.

En primer lugar, desde el menú principal, selecciona **Archivo** y a continuación **Nuevo Proyecto**. Se puede hacer más rápido con la combinación **Control + Mayúscula + N**.

Verás que aparece una ventana emergente para elegir el tipo de proyecto. Selecciona **Java** en el apartado "Categorías" y **Aplicación Java** en el apartado "Proyectos". Haz clic en **Siguiente**.

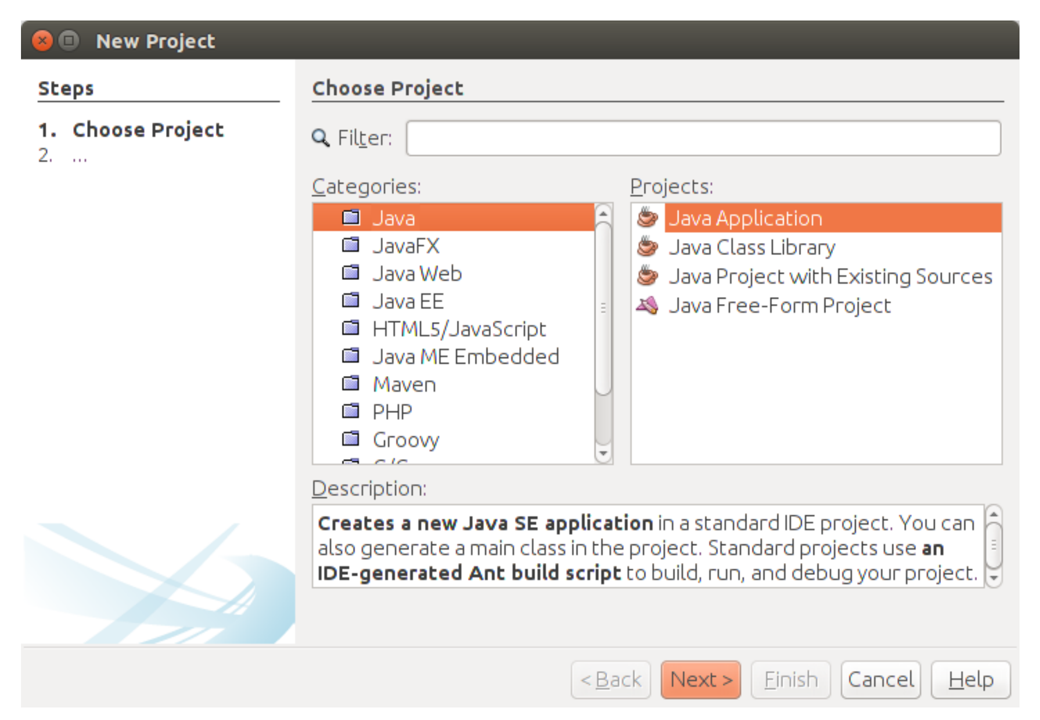


Figura B.3: Nuevo proyecto en Java

Ahora es necesario darle el nombre al proyecto, lo puedes llamar por ejemplo **Saludo**. Recuerda no utilizar caracteres especiales ni signos de puntuación. Se creará una carpeta con el mismo nombre que el proyecto que contendrá todos los ficheros necesarios. Asegúrate de que la casilla **Crear la clase principal** está marcada. Haz clic en **Terminar**.

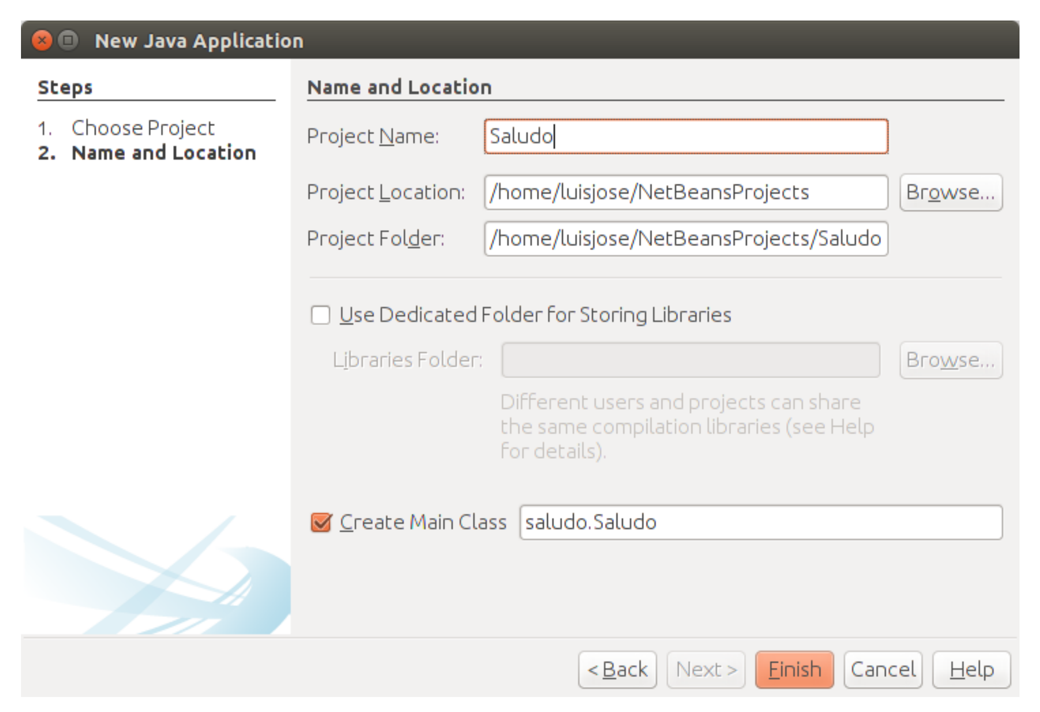


Figura B.4: Proyecto "Saludo"

Verás que se ha creado el "esqueleto" de la aplicación. La clase principal se llama `Saludo` y contiene el `main` que será el cuerpo principal del programa. Observa que en la parte de la izquierda, donde están todos los proyectos que se van realizando, puedes ver la estructura de carpetas y ficheros. El código fuente de nuestra aplicación está en el fichero `Saludo.java`. Escribe dentro del `main` la siguiente línea:

```
System.out.println("¡Hola mundo!");
```

Para guardar, compilar y ejecutar basta con hacer clic en el botón verde con el icono de *Play*, o bien pulsar la tecla **F6**.

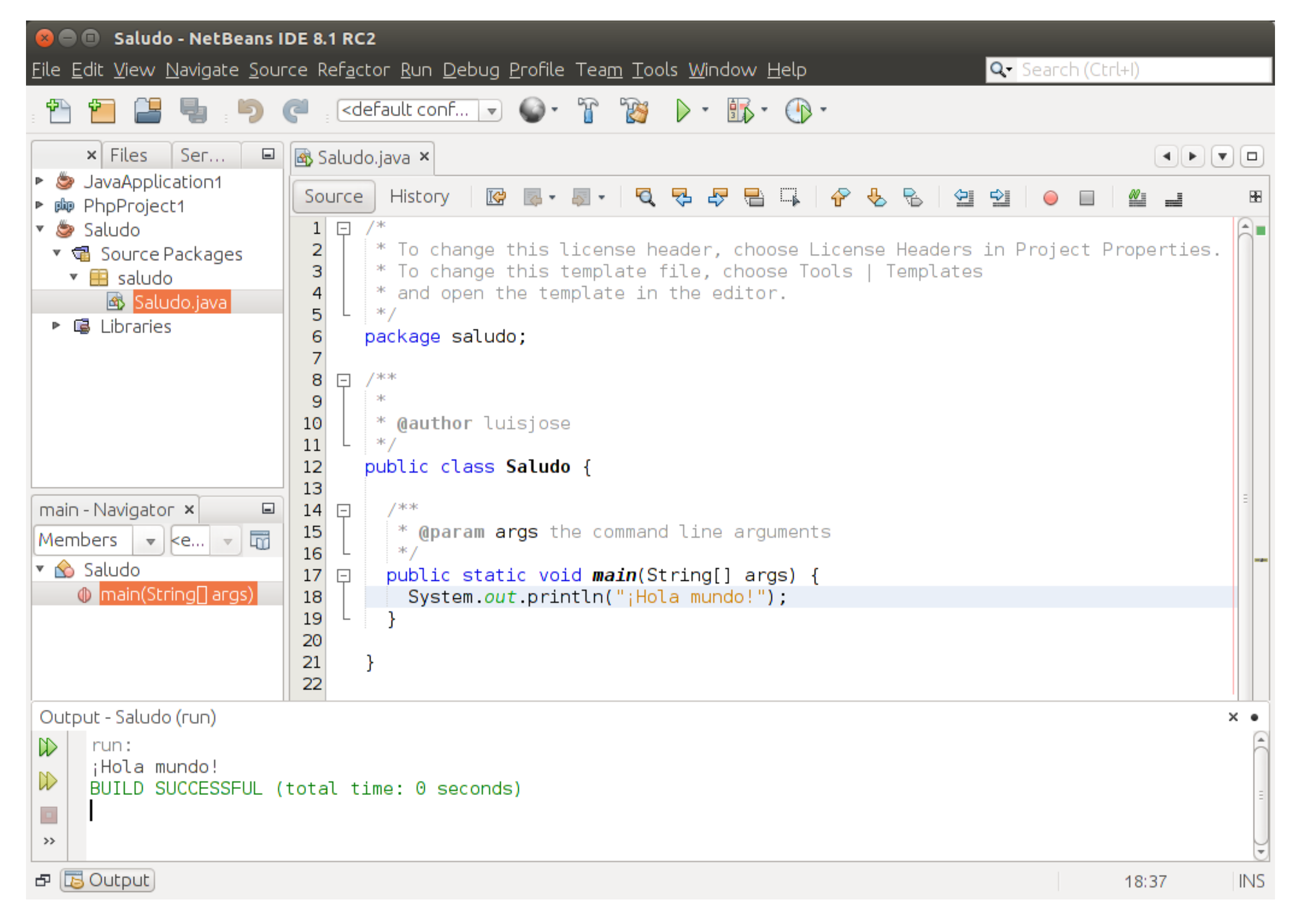


Figura B.5: Ejecución del proyecto

Si todo ha salido bien, verás el mensaje `¡Hola mundo!` en la ventana de terminal integrada de **Netbeans** que hay justo debajo del editor.

Depuración

El entorno **Netbeans** permite la depuración paso a paso de un programa. En ocasiones es complicado encontrar y corregir los fallos de una aplicación y ayuda mucho tener la posibilidad de ir ejecutando un programa línea a línea y ver cómo van cambiando las variables.

Para ilustrar la depuración paso a paso vamos a utilizar un programa de ejemplo del Capítulo 7: Arrays, en concreto usaremos el código de `Array03.java`.

Crea un nuevo proyecto de Java como se ha indicado en el apartado anterior y llámalo **Array03**. A continuación, copia el contenido del `main` del fichero `Array03.java` en tu proyecto. Ten cuidado de no borrar la línea `package array03;`.

Comenzaremos la ejecución paso a paso del programa pulsando de manera sucesiva la tecla **F7**.

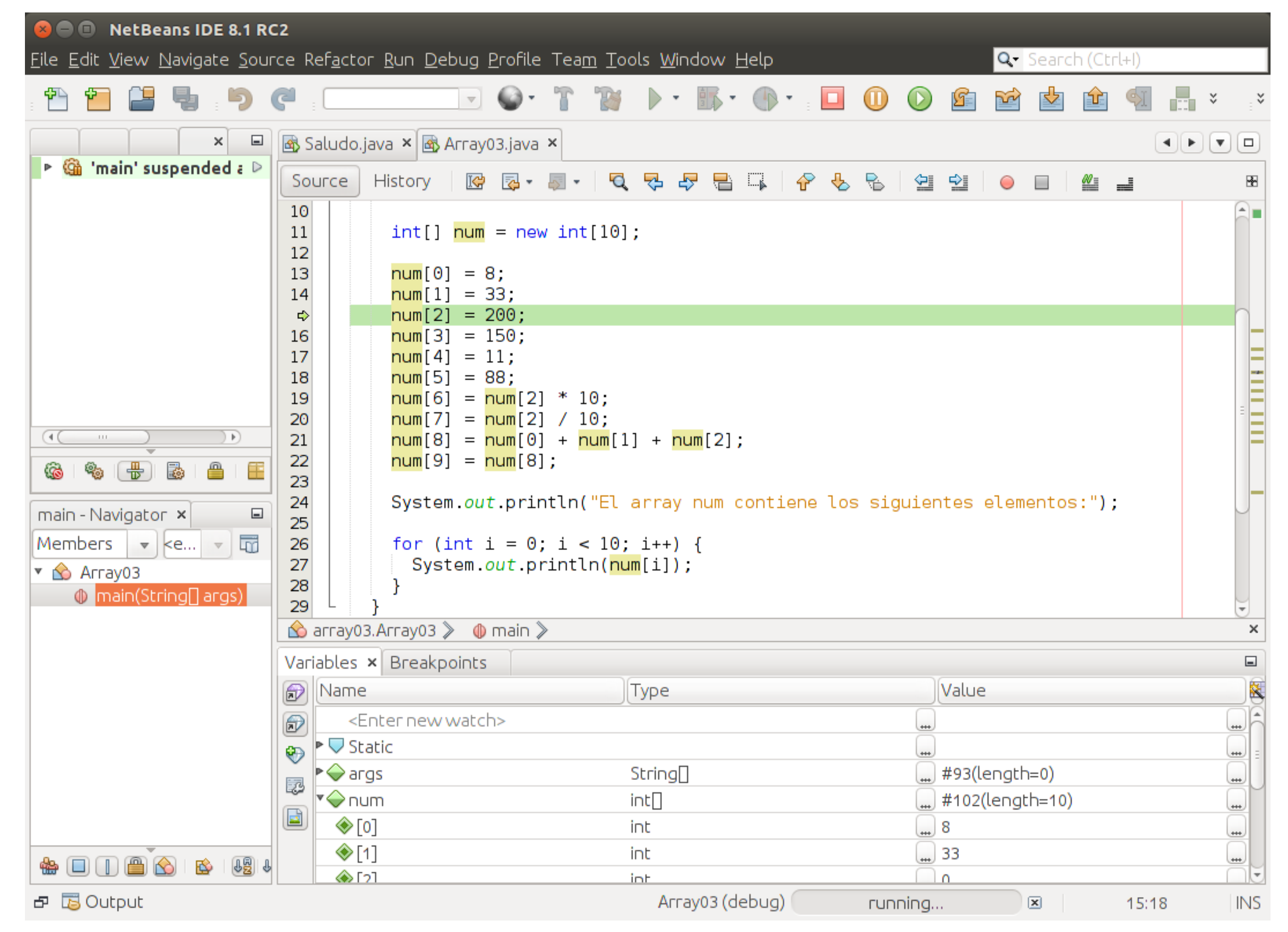


Figura B.6: Ejecución paso a paso

Observa cómo **Netbeans** resalta una línea, la ejecuta, se detiene y espera que sigamos pulsando **F7**. En la parte de abajo ha aparecido una nueva ventana que resulta tremendamente útil. Se trata de una zona donde puedes ver en tiempo real el valor que tiene cada variable (por supuesto se incluyen los arrays).

Ve pulsando **F7** hasta colocarte en la línea siguiente:

```
num[2] = 200;
```

En la ventana inferior, donde aparecen todos los datos, despliega el array `num`, verás que los elementos 0 y 1 ya tienen los valores 8 y 33 respectivamente. Si vuelves a pulsar **F7** verás que ahora, en la posición 2 se ha almacenado el 200.

Para salir del modo "paso a paso" y terminar de ejecutar el programa, pulsa **F5**.

Además de la ejecución "paso a paso" también resultan muy útiles los puntos de ruptura o *breakpoints*. Se trata de marcas, a modo de señales de STOP, donde indicamos pausas en la ejecución del programa. La depuración de un programa de varios cientos o miles de líneas puede ser muy tediosa si se ejecuta cada línea paso a

paso con **F7**. Los *breakpoints* permiten parar la ejecución en líneas concretas donde queremos ver con más detalle qué está sucediendo en nuestro programa.

Vamos a colocar varios *breakpoints* en nuestro código. Haz clic sobre las líneas 15, 20 y 24 (justo encima del número de línea). Observa que estas líneas quedan marcadas con un pequeño cuadradito.

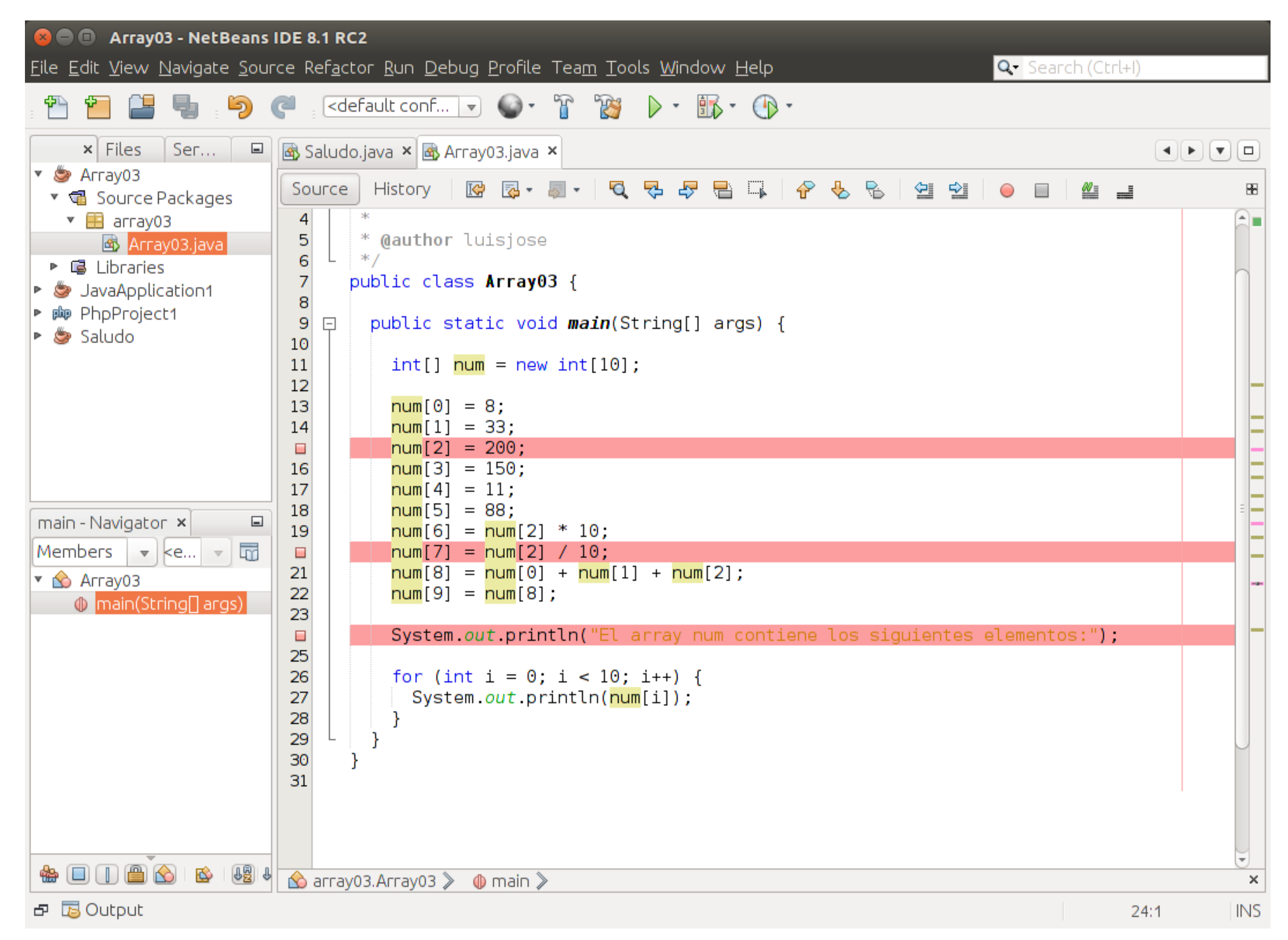


Figura B.7: *Breakpoints*

Pulsa la combinación de teclas **Control + F5**. Observa cómo el programa empieza a ejecutarse con normalidad y se para justo en la línea 15, donde está el primer *breakpoint*. Pulsando ahora **F5**, el programa continúa ejecutándose hasta que se para en la línea 20, donde está el segundo *breakpoint*. Otra pulsación de **F5** hace que continúe la ejecución hasta la línea 24. Una última pulsación de esta tecla hace que el programa continúe la ejecución hasta el final.

Apéndice C. Caracteres especiales

Líneas para tablas

─ ━ │ ┃ ┄ ┅ ┆ ┇ ┈ ┉ ┊ ┋ ┌ ┏ ┐ ┓ └ ┗ ┘ ┛ ├ ┣ ┤ ┫ ┬ ┳ ┴ ┻ ┼ ╋ ═ ║ ╔ ╗ ╚ ╝ ╠ ╣ ╦ ╩ ╬ ╭ ╮ ╯ ╰

Bloques

█ ░ ▒ ▓ ■ □ ▢ ▤ ▥ ▦ ▧ ▨ ◧ ◨ ◩ ◪ ◫ ◰ ◱ ◲ ◳ ⊞ ⊟ ⊠ ⊡

Figuras de ajedrez

♔ ♕ ♖ ♗ ♘ ♙ ♚ ♛ ♜ ♝ ♞ ♟

Círculos

◉ ○ ◌ ◍ ◎ ● ◐ ◑ ◒ ◓ ◦ ◯ ⊕ ⊖ ⊗ ⊘ ⊙ ⊛ ⊜ ⊝ ⊖ ⊕ ⦶ Ø O Q ⊙ ⌀

Flechas

← ↑ → ↓ ↔ ↕ ↖ ↗ ↘ ↙ ↚ ↛ ↜ ↝ ↞ ↟ ↠ ↡ ↢ ↣ ↤ ↥ ↦ ↧ ↨ ↩ ↪ ↫ ↬ ↭ ↮ ↯ ↰ ↱ ↲ ↳ ↴ ↵
↶ ↷ ↸ ↹ ↺ ↻ ↼ ↽ ↾ ↿ ⇀ ⇁ ⇂ ⇃ ⇄ ⇅ ⇆ ⇇ ⇈ ⇉ ⇊ ⇋ ⇌ ⇍ ⇎ ⇏ ⇐ ⇑ ⇒ ⇓ ⇔ ⇕ ⇖ ⇗ ⇘ ⇙ ⇚ ⇛
⇜ ⇝ ⇞ ⇟ ⇠ ⇡ ⇢ ⇣ ⇤ ⇥ ⇦ ⇧ ⇨ ⇩ ⇪ ⇫ ⇬ ⇭ ⇮ ⇯ ⇰ ⇱ ⇲ ⇳ ⇴ ⇵ ⇶ ⇷ ⇸ ⇹ ⇺ ⇻ ⇼ ⇽ ⇾ ⇿ ➔ ➘
➙ ➚ ➛ ➜ ➝ ➞ ➟ ➠ ➡ ➢ ➣ ➤ ➥ ➦ ➧ ➨ ➩ ➪ ➫ ➬ ➭ ➮ ➯ ➱ ➲ ➳ ➴ ➵ ➶ ➷ ➸ ➹ ➺ ➻ ➼ ➽ ➾

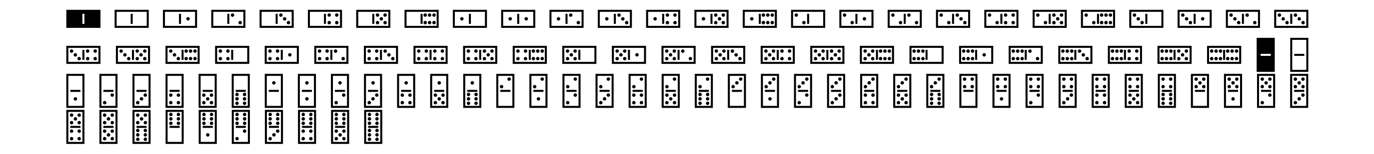
Números en círculos

❶ ❷ ❸ ❹ ❺ ❻ ❼ ❽ ❾ ❿ ➀ ➁ ➂ ➃ ➄ ➅ ➆ ➇ ➈ ➉ ➊ ➋ ➌ ➍ ➎ ➏ ➐ ➑ ➒ ➓

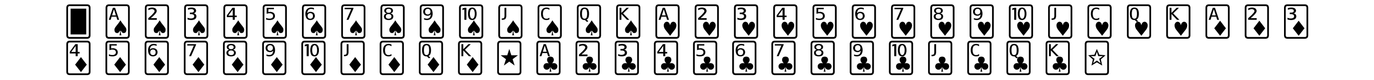
Dados

⚀ ⚁ ⚂ ⚃ ⚄ ⚅

Fichas de dominó



Cartas



Caras

🐭 🐮 🐱 🐵 😀 😁 😂 😃 😄 😅 😆 😇 😈 😉 😊 😋 😌 ☹ ☺ 😍 😎 😏 😐 😑 😒 😓 😔 😕 😖 😗 😘 😙 😚 😛 😜 😝 😞 😟 😠 😡 😢 😣 😥 😦 😧 😨 😩 😪 😫 😭 😮 😯 😰 😱 😲 😳 😴 😵 😶 😷 😸 😹 😺 😻 😼 😽 😾 😿 🙀

Horóscopo

♈ ♉ ♊ ♋ ♌ ♍ ♎ ♏ ♐ ♑ ♒ ♓

Miscelánea

⌨ ☎ ☏ ☔ ☀ ☁ ☃ ☄ ★ ☆ ☕ ☘ ☠ ☢ ☣ ☤ ☥ ☦ ☧ ☨ ☩ ☫ ☭ ☮ ☯ ☼ ☾ ♁ ♂ ※ ⁕ ⁜ Ω ♻ ⚐ ⚑ ⚒ ⚓ ⚔ ⚕ ⚖ ⚗ ⚘ ⚙ ⚚ ⚛ ⚜ ⚠ ⚡ ⛀ ⛁ ⛂ ⛃ ✂ ✇ ✈ ✉ ✌ ✍ ✎ ✑ ✒ ✓ ✔ ✕ ✖ ✙ ✚ ✛ ✜ ✝ ✞ ✟ ✠ ✡ ✢ ✣ ✤ ✥ ✦ ✧ ✩ ✪ ✫ ✬ ✭ ✮ ✯ ✰ ✱ ✲ ✳ ✴ ✵ ✶ ✷ ✸ ✹ ❀

Apéndice D. Referencias

Java

- Aprende Java con Ejercicios[8]
- Ejemplos y soluciones a los ejercicios de “Aprende Java con Ejercicios”[9]
- Google Java Style Guide[10]

Git y GitHub

- Git y GitHub. Guía de Supervivencia.[11]

HTML

- w3schools.com[12]
- Getting to Know HTML[13]

Caracteres Unicode

- Unicode 9.0 Character Code Charts[14]

---

[8] https://leanpub.com/aprendejava
[9] https://github.com/LuisJoseSanchez/aprende-java-con-ejercicios
[10] https://google.github.io/styleguide/javaguide.html
[11] https://leanpub.com/gitygithub/
[12] http://www.w3schools.com/
[13] http://learn.shayhowe.com/html-css/getting-to-know-html/
[14] http://www.unicode.org/charts/

Apéndice E. Soluciones a los ejercicios

A continuación se presentan las soluciones a los ejercicios del libro. No son las únicas posibles ni necesariamente las mejores.

Estas soluciones se actualizan con cada nueva edición del libro. La versión más actualizada está en este repositorio de GitHub[15]. Si detectas cualquier error o posible mejora, te invito a hacer *pull request*[16].

[15] https://github.com/LuisJoseSanchez/aprende-java-con-ejercicios

[16] Si no sabes lo que significa *pull request*, te recomiendo leer mi libro Git y GitHub. Guía de Supervivencia. que puedes descargar de forma gratuita.

¡Hola mundo! - Salida de datos por pantalla

Ejercicio 1

Fichero: S01Ejercicio01.java

```
/**
 * 1. Salida por pantalla
 *
 * 1. Escribe una programa que muestre tu nombre por pantalla.
 *
 * @author Luis José Sánchez
 */
public class S01Ejercicio01 {
  public static void main(String[] args) {
    System.out.println("Luis José Sánchez González");
  }
}
```

Ejercicio 2

Fichero: S01Ejercicio02.java

```
/**
 * 1. Salida por pantalla
 *
 * 2. Modifica el programa anterior para que además se muestre tu
 *    dirección y tu número de teléfono. Asegúrate de que los
 *    datos se muestran en líneas separadas.
 *
 * @author Luis José Sánchez
 */
public class S01Ejercicio02 {
  public static void main(String[] args) {
    System.out.println("Luis José Sánchez González");
    System.out.println("Larios, 180 - Málaga");
    System.out.println("Tel: 555 12 34 56");
  }
}
```

Ejercicio 3

Fichero: S01Ejercicio03.java

```java
/**
 * 1. Salida por pantalla.
 *
 * Solución al ejercicio 3.
 *
 * "Aprende Java con Ejercicios" (https://leanpub.com/aprendejava)
 *
 * @author Luis José Sánchez
 */
public class S01Ejercicio03 {
  public static void main(String[] args) {
    System.out.printf("%-10s %s\n", "computer", "ordenador");
    System.out.printf("%-10s %s\n", "student", "alumno\\a");
    System.out.printf("%-10s %s\n", "cat", "gato");
    System.out.printf("%-10s %s\n", "penguin", "pingüino");
    System.out.printf("%-10s %s\n", "machine", "máquina");
    System.out.printf("%-10s %s\n", "nature", "naturaleza");
    System.out.printf("%-10s %s\n", "light", "luz");
    System.out.printf("%-10s %s\n", "green", "verde");
    System.out.printf("%-10s %s\n", "book", "libro");
    System.out.printf("%-10s %s\n", "pyramid", "pirámide");
  }
}
```

Ejercicio 4

Fichero: S01Ejercicio04.java

```java
/**
 * 1. Salida por pantalla
 *
 * 4. Escribe un programa que muestre tu horario de clase. Puedes usar
 *    espacios o tabuladores para alinear el texto.
 *
 * @author Luis José Sánchez
 */
public class S01Ejercicio04 {
  public static void main(String[] args) {
    System.out.println("Lunes\tMartes\tMiérc.\tJueves.\tViernes");
    System.out.println("======\t=======\t======\t=======\t=======");
    System.out.println("PROG\tPROG\tPROG\tPROG \tSIN");
    System.out.println("PROG\tPROG\tPROG\tPROG\tSIN");
    System.out.println("ED\tSIN\tSIN\tLM\tBDATO");
    System.out.println("FOL\tSIN\tSIN\tLM\tBDATO");
    System.out.println("FOL\tBDATO\tED\tBDATO\tED");
```

```
    System.out.println("FOL\tBDATO\tED\tBDATO\tED");
  }
}
```

Ejercicio 5

Fichero: S01Ejercicio05.java

```
/**
 * 1. Salida por pantalla
 *
 * 5. Modifica el programa anterior añadiendo colores. Puedes mostrar
 *    cada asignatura de un color diferente.
 *
 * @author Luis José Sánchez
 */
public class S01Ejercicio05 {
  public static void main(String[] args) {
    String rojo = "\033[31m";
    String verde = "\033[32m";
    String naranja = "\033[33m";
    String azul = "\033[34m";
    String morado = "\033[35m";
    String celeste = "\033[36m";
    String blanco = "\033[37m";

    System.out.println(naranja + "Lunes\tMartes\tMiérc.\tJueves\tViernes");
    System.out.println("======\t=======\t======\t======\t=======");
    System.out.println(verde + "PROG\tPROG\tPROG\tPROG" + rojo + "\tSIN");
    System.out.println(verde + "PROG\tPROG\tPROG\tPROG" + rojo + "\tSIN");
    System.out.println(celeste + "ED" + rojo + "\tSIN\tSIN" + blanco + "\tLM" + morado + "\tBD\
ATO");
    System.out.println(naranja + "FOL" + rojo + "\tSIN\tSIN" + blanco + "\tLM" + morado + "\tB\
DATO");
    System.out.println(naranja + "FOL" + morado + "\tBDATO" + celeste + "\tED" + morado + "\tB\
DATO" + celeste + "\tED");
    System.out.println(naranja + "FOL" + morado + "\tBDATO" + celeste + "\tED" + morado + "\tB\
DATO" + celeste + "\tED");
  }
}
```

Ejercicio 6

Fichero: S01Ejercicio06.java

```
/**
 * 1. Salida por pantalla
 *
 * 6. Escribe un programa que pinte por pantalla una pirámide rellena a
 *    base de asteriscos. La base de la pirámide debe estar formada por
 *    9 asteriscos.
 *
 * @author Luis José Sánchez
 *
 */
public class S01Ejercicio06 {
  public static void main(String[] args) {
    System.out.println("    *");
    System.out.println("   ***");
    System.out.println("  *****");
    System.out.println(" *******");
    System.out.println("*********");
  }
}
```

Ejercicio 7

Fichero: S01Ejercicio07.java

```
/**
 * 1. Salida por pantalla
 *
 * 7. Igual que el programa anterior, pero esta vez la pirámide estará
 *    hueca (se debe ver únicamente el contorno hecho con asteriscos).
 *
 * @author Luis José Sánchez
 */
public class S01Ejercicio07 {
  public static void main(String[] args) {
    System.out.println("\033[34m"); // pintamos en azul

    System.out.println("    *");
    System.out.println("   * *");
    System.out.println("  *   *");
    System.out.println(" *     *");
    System.out.println("*********");

    System.out.println("\033[37m"); // volvemos al blanco
  }
}
```

Ejercicio 8

Fichero: S01Ejercicio08.java

```java
/**
 * 1. Salida por pantalla
 *
 * 8. Igual que el programa anterior, pero esta vez la pirámide debe
 *    aparecer invertida, con el vértice hacia abajo.
 *
 * @author Luis José Sánchez
 */

public class S01Ejercicio08 {
  public static void main(String[] args) {
    System.out.println("\033[34m"); // pintamos en azul

    System.out.println("*********");
    System.out.println(" *     *");
    System.out.println("  *   *");
    System.out.println("   * *");
    System.out.println("    *");

    System.out.println("\033[37m"); // volvemos al blanco
  }
}
```

Ejercicio 9

Fichero: S01Ejercicio09.java

```java
/**
 * 1. Salida por pantalla
 *
 * 9. Escribe un programa que pinte por pantalla alguna escena - el campo,
 *    la habitación de una casa, un aula, etc. - o algún objeto animado o
 *    inanimado - un coche, un gato, una taza de café, etc. Ten en cuenta
 *    que puedes utilizar caracteres como *, +, <, #, @, etc. o incluso
 *    caracteres Unicode. ¡Échale imaginación!
 *
 * @author Luis José Sánchez
 *
 */
public class S01Ejercicio09 {
  public static void main(String[] args) {
```

```
    System.out.println("\n                        □□");
    System.out.println("    APRENDE JAVA      □□");
    System.out.println("   CON EJERCICIOS       □□\n");
    System.out.println("  Luis José Sánchez  □□□□□□□□□□□□");
    System.out.println("                     □□□□□□□□  □□");
    System.out.println("                     □□□□□□□□□□□□");
    System.out.println("                     □□□□□□□□");
    System.out.println("□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□");
    System.out.println("□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□");
    System.out.println("□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□");
  }
}
```

Ejercicio 10

Fichero: S01Ejercicio09.java

```
/**
 * 1. Salida por pantalla
 *
 * 10. Mejora el ejercicio anterior añadiéndole colores.
 *
 * @author Luis José Sánchez
 */
public class S01Ejercicio10 {
  public static void main(String[] args) {
    System.out.println("\033[31m");
    System.out.println("                        □□");
    System.out.println("    APRENDE JAVA      □□");
    System.out.println("   CON EJERCICIOS       □□\n\033[33m");
    System.out.println("  Luis José Sánchez \033[34m □□□□□□□□□□□□");
    System.out.println("                     □□□□□□□□  □□");
    System.out.println("                     □□□□□□□□□□□□");
    System.out.println("                     □□□□□□□□");
    System.out.println("□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□");
    System.out.println("□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□");
    System.out.println("□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□");
    System.out.println("\033[37m"); // volvemos al blanco
  }
}
```

Variables

Ejercicio 1

Fichero: S02Ejercicio01.java

```
/**
 *
 * 2. Variables
 *
 * 1. Escribe un programa en el que se declaren las variables enteras x
 *    e y. Asignales los valores 144 y 999 respectivamente. A continuación,
 *    muestra por pantalla el valor de cada variable, la suma, la resta,
 *    la división y la multiplicación.
 *
 * @author Luis José Sánchez
 */
public class S02Ejercicio01 {
  public static void main(String[] args) {
    int x = 144;
    int y = 999;

    System.out.println("x = " + x);
    System.out.println("y = " + y);
    System.out.println("x + y = " + (x + y));
    System.out.println("x - y = " + (x - y));
    System.out.println("x / y = " + ((float)x / (float)y));
    System.out.println("x * y = " + (x * y));
  }
}
```

Ejercicio 2

Fichero: S02Ejercicio02.java

```
/**
 * 2. Variables
 *
 * 2. Crea la variable nombre y asígnale tu nombre completo. Muestra su
 *    valor por pantalla de tal forma que el resultado del programa sea
 *    el mismo que en el ejercicio 1 del capítulo 1.
 *
 * @author Luis José Sánchez
 */
public class S02Ejercicio02 {
  public static void main(String[] args) {
    String nombre = "Luis José Sánchez González";
    System.out.println(nombre);
  }
}
```

Ejercicio 3

Fichero: S02Ejercicio03.java

```
/**
 * 2. Variables
 *
 * 3. Crea las variables nombre, direccion y telefono y asígnale los
 *    valores correspondientes. Muestra los valores de esas variables
 *    por pantalla de tal forma que el resultado del programa sea el
 *    mismo que en el ejercicio 2 del capítulo 1.
 *
 * @author Luis José Sánchez
 */
public class S02Ejercicio03 {
  public static void main(String[] args) {
    String nombre = "Luis José Sánchez González";
    String direccion = "Larios, 180 - Málaga";
    String telefono = "Tel: 555 12 34 56";
    System.out.println(nombre);
    System.out.println(direccion);
    System.out.println(telefono);
  }
}
```

Ejercicio 4

Fichero: S02Ejercicio04.java

```java
/**
 * 2. Variables
 *
 * 4. Realiza un conversor de euros a pesetas. La cantidad en euros que
 *    se quiere convertir deberá estar almacenada en una variable.
 *
 * @author Luis José Sánchez
 */
public class S02Ejercicio04 {
  public static void main(String[] args) {
    double euros = 6.00;
    int pesetas = (int) (euros * 166.386);

    System.out.print(euros + " euros son " + pesetas + " pesetas.");
  }
}
```

Ejercicio 5

Fichero: S02Ejercicio05.java

```java
/**
 * 2. Variables
 *
 * 5. Realiza un conversor de pesetas a euros. La cantidad en pesetas
 *    que se quiere convertir deberá estar almacenada en una variable.
 *
 * @author Luis José Sánchez
 */
public class S02Ejercicio05 {
  public static void main(String[] args) {
    int pesetas = 10000;
    double euros = pesetas / 166.386;

    System.out.println(pesetas + " pesetas son " + euros + " euros.");
    // Para mostrar dos decimales se puede usar "printf"
    // en lugar de "print" o "println"
    System.out.printf("%d pesetas son %.2f euros.\n", pesetas, euros);
  }
}
```

Ejercicio 6

Fichero: S02Ejercicio06.java

```
/**
 * 2. Variables
 *
 * 6. Escribe un programa que calcule el total de una factura a partir
 *    de la base imponible (precio sin IVA). La base imponible estará
 *    almacenada en una variable.
 *
 * @author Luis José Sánchez
 */
public class S02Ejercicio06 {
  public static void main(String[] args) {

    double baseImponible = 22.75;

    System.out.printf("Base imponible %8.2f\n", baseImponible);
    System.out.printf("IVA            %8.2f\n", (baseImponible * 0.21));
    System.out.printf("-----------------------\n");
    System.out.printf("Total          %8.2f\n", (baseImponible * 1.21));
  }
}
```

Ejercicio 7

Fichero: S02Ejercicio07.java

```
public class S02Ejercicio07 {
  public static void main(String[] args) {
    char primeraLetra = 'a';
    char ultimaLetra = 'z';
    String palabra = "abecedario";

    System.out.println(primeraLetra + "" + ultimaLetra + palabra);
  }
}
```

Ejercicio 8

Fichero: S02Ejercicio08.java

```
public class S02Ejercicio08 {
  public static void main(String[] args) {
    char l1 = 's';
    char l2 = 'a';
    char l3 = 'l';
    char l4 = 'u';
    char l5 = 'd';

    // La siguiente línea da error porque el resultado de sumar variables
    // de tipo carácter es un número entero.
    // String palabra = l1 + l2 + l3 + l4 + l5;

    // Se soluciona concatenando al principio la cadena vacía.
    String palabra = "" + l1 + l2 + l3 + l4 + l5;

    System.out.println(palabra);
  }
}
```

Lectura de datos desde teclado

Ejercicio 1

Fichero: S03Ejercicio01.java

```
/**
 * 3. Lectura de datos desde teclado
 *
 * 1. Realiza un programa que pida dos números y que luego muestre el
 *    resultado de su multiplicación.
 *
 * @author Luis José Sánchez
 */

public class S03Ejercicio01 {
  public static void main(String[] args) {
    int x;
    int y;
    String linea;

    System.out.print("Por favor, introduce el primer número: ");
    linea = System.console().readLine();
    x = Integer.parseInt( linea );
    System.out.print("Introduce el segundo número: ");
    linea = System.console().readLine();
    y = Integer.parseInt( linea );

    System.out.println(x + "*" + y + "=" + (x * y));
  }
}
```

Ejercicio 2

Fichero: S03Ejercicio02.java

```
/**
 * 3. Lectura de datos desde teclado
 *
 * 2. Realiza un conversor de euros a pesetas. La cantidad de euros que
 *    se quiere convertir debe ser introducida por teclado.
 *
 * @author Luis José Sánchez
 */

public class S03Ejercicio02 {
  public static void main(String[] args) {

    System.out.print("Por favor, introduce la cantidad de euros que quieres convertir: ");
    double euros = Double.parseDouble(System.console().readLine());

    int pesetas = (int) (euros * 166.386);

    System.out.print(euros + " euros son " + pesetas + " pesetas.");
  }
}
```

Ejercicio 3

Fichero: S03Ejercicio03.java

```
/**
 * 3. Lectura de datos desde teclado
 *
 * 3. Realiza un conversor de pesetas a euros. La cantidad de pesetas
 *    que se quiere convertir debe ser introducida por teclado.
 *
 * @author Luis José Sánchez
 */

public class S03Ejercicio03 {
  public static void main(String[] args) {

    System.out.print("Por favor, introduzca la cantidad de pesetas que quiere convertir: ");
    int pesetas = Integer.parseInt(System.console().readLine());

    double euros = pesetas / 166.386;

    System.out.printf("%d pesetas son %.2f euros.", pesetas, euros);
  }
}
```

Ejercicio 4

Fichero: S03Ejercicio04.java

```
/**
 * 3. Lectura de datos desde teclado
 *
 * 4. Escribe un programa que sume, reste, multiplique y divida dos
 *    números introducidos por teclado.
 *
 * @author Luis José Sánchez
 */

public class S03Ejercicio04 {
  public static void main(String[] args) {
    int x;
    int y;
    String linea;

    System.out.print("Por favor, introduzca el primer número: ");
    linea = System.console().readLine();
    x = Integer.parseInt( linea );
    System.out.print("Introduzca el segundo número: ");
    linea = System.console().readLine();
    y = Integer.parseInt( linea );

    System.out.println("x = " + x);
    System.out.println("y = " + y);
    System.out.println("x + y = " + (x + y));
    System.out.println("x - y = " + (x - y));
    System.out.println("x / y = " + (x / y));
    System.out.println("x * y = " + (x * y));
  }
}
```

Ejercicio 5

Fichero: S03Ejercicio05.java

```
/**
 * 3. Lectura de datos desde teclado
 *
 * 5. Escribe un programa que calcule el área de un rectángulo.
 *
 * @author Luis José Sánchez
 */
public class S03Ejercicio05 {
  public static void main(String[] args) {

    System.out.println("Área de un rectángulo");
    System.out.print("Por favor, introduzca la longitud de la base (cm): ");
    double base = Double.parseDouble(System.console().readLine());
    System.out.print("Introduzca la altura (cm): ");
    double altura = Double.parseDouble(System.console().readLine());
    System.out.println("El área del rectángulo es " + (base * altura) + " cm²");
  }
}
```

Ejercicio 6

Fichero: S03Ejercicio06.java

```
/**
 * 3. Lectura de datos desde teclado
 *
 * 6. Escribe un programa que calcule el área de un triángulo.
 *
 * @author Luis José Sánchez
 */
public class S03Ejercicio06 {
  public static void main(String[] args) {

    System.out.println("Área de un triángulo");
    System.out.print("Por favor, introduzca la longitud de la base (cm): ");
    double base = Double.parseDouble(System.console().readLine());
    System.out.print("Inroduzca la altura (cm): ");
    double altura = Double.parseDouble(System.console().readLine());
    System.out.println("El área del triángulo es " + (base * altura)/2 + " cm²");
  }
}
```

Ejercicio 7

Fichero: S03Ejercicio07.java

```
/**
 * 3. Lectura de datos desde teclado
 *
 * 7. Escribe un programa que calcule el total de una factura a partir
 *    de la base imponible.
 *
 * @author Luis José Sánchez
 */

public class S03Ejercicio07 {
  public static void main(String[] args) {

    System.out.print("Por favor, introduzca la base imponible (precio del artículo sin IVA): "\
);
    double baseImponible = Double.parseDouble(System.console().readLine());

    System.out.printf("Base imponible %8.2f\n", baseImponible);
    System.out.printf("IVA            %8.2f\n", (baseImponible * 0.21));
    System.out.printf("-----------------------\n");
    System.out.printf("Total          %8.2f\n", (baseImponible * 1.21));
  }
}
```

Ejercicio 8

Fichero: S03Ejercicio08.java

```
/**
 * 3. Lectura de datos desde teclado
 *
 * 8. Escribe un programa que calcule el salario semanal de un empleado
 *    en base a las horas trabajadas, a razón de 12 euros la hora.
 *
 * @author Luis José Sánchez
 */

public class S03Ejercicio08 {
  public static void main(String[] args) {

    System.out.print("Por favor, introduzca el número de horas trabajadas durante la semana: "\
);
    int horasTrabajadas = Integer.parseInt(System.console().readLine());
    System.out.println("Su salario semanal es de " + (horasTrabajadas * 12) + " euros.");
  }
}
```

Ejercicio 9

Fichero: S03Ejercicio09.java

```
/**
 * 3. Lectura de datos desde teclado
 *
 * 9. Escribe un programa que calcule el volumen de un cono según la
 *    fórmula V = (1/3)πr^2h
 *
 * @author Luis José Sánchez
 */
public class S03Ejercicio09 {
  public static void main(String[] args) {

    final double PI = 3.141592654;
    // En lugar de definir una constante se podría usar
    // la constante predefinida Math.PI
    System.out.println("Volumen de un cono");
    System.out.print("Por favor, introduzca la altura (cm): ");
    double h = Double.parseDouble(System.console().readLine());
    System.out.print("Introduzca el radio (cm): ");
    double r = Double.parseDouble(System.console().readLine());
    double v = (1.0/3.0) * PI * r * r * h;
    System.out.println("El volumen del cono es de " +  v + " cm³");
  }
}
```

Ejercicio 10

Fichero: S03Ejercicio10.java

```
/**
 * 3. Lectura de datos desde teclado
 *
 * 10. Realiza un conversor de Mb a Kb.
 *
 * @author Luis José Sánchez
 */

public class S03Ejercicio10 {
  public static void main(String[] args) {

    System.out.print("Por favor, introduzca el número de Mb: ");
    double mb = Double.parseDouble(System.console().readLine());
```

```
    System.out.println(mb + "Mb son " +  (mb * 1024) + "Kb.");
  }
}
```

Ejercicio 11

Fichero: S03Ejercicio11.java

```
/**
 * 3. Lectura de datos desde teclado
 *
 * 11. Realiza un conversor de Kb a Mb.
 *
 * @author Luis José Sánchez
 */

public class S03Ejercicio11 {
  public static void main(String[] args) {

    System.out.print("Por favor, introduzca el número de Kb: ");
    double kb = Double.parseDouble(System.console().readLine());
    System.out.println(kb + "Kb son " +  (kb / 1024) + "Mb.");
  }
}
```

Ejercicio 12

Fichero: S03Ejercicio12.java

```
/**
 * 3. Lectura de datos desde teclado
 *
 * 12.  Realiza un programa que calcule la nota que hace falta sacar en el segundo
 *      examen de la asignatura **Programación** para obtener la media deseada.
 *      Hay que tener en cuenta que la nota del primer examen cuenta el 40% y la
 *      del segundo examen un 60%.
 *
 *      Ejemplo 1:
 *      Introduce la nota del primer examen: 7
 *      ¿Qué nota quieres sacar en el trimestre? 8.5
 *      Para tener un 8.5 en el trimestre necesitas sacar un 9.5 en el segundo examen.
 *
 *      Ejemplo 2:
 *      Introduce la nota del primer examen: 8
```

```
 *      ¿Qué nota quieres sacar en el trimestre? 7
 *      Para tener un 7 en el trimestre necesitas sacar un 6.33 en el segundo examen.
 *
 * @author Luis José Sánchez
 */

public class S03Ejercicio12 {
  public static void main(String[] args) {

    System.out.print("Introduce la nota del primer examen: ");
    double nota1 = Double.parseDouble(System.console().readLine());

    System.out.print("¿Qué nota quieres sacar en el trimestre? ");
    double notaFinal = Double.parseDouble(System.console().readLine());

    // La nota final se calcula de esta forma:
    // notaFinal = ((nota1 * 40) + (nota2 * 60)) / 100
    // Por tanto, despejando nota2 tenemos:
    // nota2 = ((notaFinal * 100) - (nota1 * 40)) / 60
    double nota2 = ((notaFinal * 100) - (nota1 * 40)) / 60;

    System.out.println("Para tener un " + notaFinal + " en el trimestre necesitas sacar un " +\
 nota2 + " en el segundo examen.");
  }
}
```

Sentencia condicional (`if` y `switch`)

Ejercicio 1

Fichero: S04Ejercicio01.java

```
/**
 * 4. Sentencia Condicional
 *
 * 1. Escribe un programa que pida por teclado un día de la semana
 *    y que diga qué asignatura toca a primera hora ese día.
 *
 * @author Luis José Sánchez
 */

public class S04Ejercicio01 {
  public static void main(String[] args) {

    String dia;

    System.out.print("Por favor, introduzca un día de la semana y le diré qué asignatura toca \
a primera hora ese día: ");
    dia = (System.console().readLine());
    dia.toLowerCase(); // convierto a minúsculas todas las letras

    switch(dia) {
      case "lunes":
        // continúa debajo
      case "martes":
        // continúa debajo
      case "miércoles":
        System.out.println("Programación");
        break;
      case "jueves":
        System.out.println("Sistemas Informáticos");
        break;
      case "viernes":
        System.out.println("Bases de Datos");
        break;
      case "sábado":
        // continúa debajo
      case "domingo":
        System.out.println("¡Ese día no hay clase!");
        break;
      default:
```

```
      System.out.println("El día introducido no es correcto.");
    }
  }
}
```

Ejercicio 2

Fichero: S04Ejercicio02.java

```
/**
 * 4. Sentencia Condicional
 *
 * 2. Realiza un programa que pida una hora por teclado y que muestre
 *    luego buenos días, buenas tardes o buenas noches según la hora.
 *    Se utilizarán los tramos de 6 a 12, de 13 a 20 y de 21 a 5
 *    respectivamente. Sólo se tienen en cuenta las horas, los minutos
 *    no se deben introducir por teclado.
 *
 * @author Luis José Sánchez
 */

public class S04Ejercicio02 {
  public static void main(String[] args) {

    System.out.print("Por favor, introduzca una hora del día (0 - 23): ");
    int hora = Integer.parseInt(System.console().readLine());

    if ((hora >= 6) && (hora <= 12)) {
      System.out.println("Buenos días");
    }

    if ((hora >= 13) && (hora <= 20)) {
      System.out.println("Buenas tardes");
    }

    if (((hora >= 21) && (hora < 24)) || ((hora <= 5) && (hora >= 0))) {
      System.out.println("Buenas noches");
    }

    if ((hora >= 24) || (hora < 0)) {
      System.out.println("La hora introducida no es correcta.");
    }
  }
}
```

Ejercicio 3

Fichero: S04Ejercicio03.java

```
/**
 * 4. Sentencia Condicional
 *
 * 3. Escribe un programa en que dado un número del 1 a 7 escriba el
 *    correspondiente nombre del día de la semana.
 *
 * @author Luis José Sánchez
 */

public class S04Ejercicio03 {
  public static void main(String[] args) {

    String dia;

    System.out.print("Por favor, introduzca un número del 1 al 7: ");
    int n = Integer.parseInt(System.console().readLine());

    switch(n) {
      case 1:
        dia = "lunes";
        break;
      case 2:
        dia = "martes";
        break;
      case 3:
        dia = "miércoles";
        break;
      case 4:
        dia = "jueves";
        break;
      case 5:
        dia = "viernes";
        break;
      case 6:
        dia = "sábado";
        break;
      case 7:
        dia = "domingo";
        break;
      default:
        dia = "Debe introducir un número del 1 al 7";
    }
```

```
    System.out.println(dia);

  }
}
```

Ejercicio 4

Fichero: S04Ejercicio04.java

```
/**
 * 4. Sentencia Condicional
 *
 * 4. Vamos a ampliar uno de los ejercicios de la relación anterior para
 *    considerar las extras. Escribe un programa que calcule el salario
 *    semanal de un trabajador teniendo en cuenta que las horas
 *    ordinarias (40 primeras horas de trabajo) se pagan a 12 euros la
 *    hora. A partir de la hora 41, se pagan a 16 euros la hora.
 *
 * @author Luis José Sánchez
 */
public class S04Ejercicio04 {
  public static void main(String[] args) {

    int sueldoSemanal;

    System.out.print("Por favor, introduzca el número de horas trabajadas durante la semana: "\
);
    int horasTrabajadas = Integer.parseInt(System.console().readLine());

    if (horasTrabajadas < 40) {
      sueldoSemanal = 12 * horasTrabajadas;
    } else {
      sueldoSemanal = (40 * 12) + ((horasTrabajadas - 40) * 16);
    }
    System.out.println("El sueldo semanal que le corresponde es de " + sueldoSemanal + " euros\
");
  }
}
```

Ejercicio 5

Fichero: S04Ejercicio05.java

```
/**
 * 4. Sentencia Condicional
 *
 * 5. Realiza un programa que resuelva una ecuación de primer grado
 *    (del tipo ax + b = 0).
 *
 * @author Luis José Sánchez
 */

public class S04Ejercicio05 {
  public static void main(String[] args) {

    System.out.println("Este programa resuelve ecuaciones de primer grado del tipo ax + b = 0"\
);
    System.out.print("Por favor, introduzca el valor de a: ");
    Double a = Double.parseDouble(System.console().readLine());
    System.out.print("Ahora introduzca el valor de b: ");
    Double b = Double.parseDouble(System.console().readLine());

    if (a == 0) {
      System.out.println("Esa ecuación no tiene solución real.");
    } else {
      System.out.println("x = " + (-b/a));
    }
  }
}
```

Ejercicio 6

Fichero: S04Ejercicio06.java

```
/**
 * 4. Sentencia Condicional
 *
 * 6. Realiza un programa que calcule el tiempo que tardará en caer un
 *    objeto desde una altura h.
 *    t = raiz(2h/g) siendo g = 9.81 m/s2
 *
 * @author Luis José Sánchez
 */
public class S04Ejercicio06 {

  // las constantes se declaran fuera del main
  final static double g = 9.81;
```

```
  public static void main(String[] args) {

    System.out.println("Cálculo del tiempo de caída de un objeto.");
    System.out.print("Por favor, introduzca la altura (en metros) desde la que cae el objeto: \
");
    Double h = Double.parseDouble(System.console().readLine());

    double s = Math.sqrt(2*h/g);

    System.out.printf("El objeto tarda %.2f segundos en caer.\n", s);
  }
}
```

Ejercicio 7

Fichero: S04Ejercicio07.java

```
/**
 * 4. Sentencia Condicional
 *
 * 7. Realiza un programa que calcule la media de tres notas.
 *
 * @author Luis José Sánchez
 */
public class S04Ejercicio07 {

  public static void main(String[] args) {

    System.out.println("Este programa calcula la media de tres notas.");

    System.out.print("Por favor, introduzca la primera nota: ");
    Double nota1 = Double.parseDouble(System.console().readLine());

    System.out.print("Ahora introduzca la segunda nota: ");
    Double nota2 = Double.parseDouble(System.console().readLine());

    System.out.print("Por último introduzca la tercera nota: ");
    Double nota3 = Double.parseDouble(System.console().readLine());

    double media = (nota1 + nota2 + nota3) / 3;

    System.out.printf("La media es %.2f\n", media);
  }
}
```

Ejercicio 8

Fichero: S04Ejercicio08.java

```java
/**
 * 4. Sentencia Condicional
 *
 * 8. Amplía el programa anterior para que diga la nota del boletín
 *    (insuficiente, suficiente, bien, notable o sobresaliente).
 *
 * @author Luis José Sánchez
 */
public class S04Ejercicio08 {

  public static void main(String[] args) {

    System.out.println("Este programa calcula la media de tres notas.");

    System.out.print("Por favor, introduzca la primera nota: ");
    Double nota1 = Double.parseDouble(System.console().readLine());

    System.out.print("Ahora introduzca la segunda nota: ");
    Double nota2 = Double.parseDouble(System.console().readLine());

    System.out.print("Por último introduzca la tercera nota: ");
    Double nota3 = Double.parseDouble(System.console().readLine());

    double media = (nota1 + nota2 + nota3) / 3;

    System.out.printf("La media es %.2f\n", media);

    if (media < 5) {
      System.out.print("Insuficiente");
    }

    if ((media >= 5) && (media < 6)) {
      System.out.print("Suficiente");
    }

    if ((media >= 6) && (media < 7)) {
      System.out.print("Bien");
    }

    if ((media >= 7) && (media < 9)) {
      System.out.print("Notable");
    }
```

```
    if (media >= 9) {
      System.out.print("Sobresaliente");
    }
  }
}
```

Ejercicio 9

Fichero: S04Ejercicio09.java

```
/**
 * 4. Sentencia Condicional
 *
 * 9. Realiza un programa que resuelva una ecuación de segundo grado
 *    (del tipo ax2 + bx + c = 0).
 *
 * @author Luis José Sánchez
 */
public class S04Ejercicio09 {

  public static void main(String[] args) {

    double x1, x2;

    System.out.println("Este programa resuelve ecuaciones de segundo grado.");
    System.out.println("ax^2 + bx + c = 0");

    System.out.println("Por favor, introduzca los valores.");

    System.out.print("a = ");
    double a = Double.parseDouble(System.console().readLine());

    System.out.print("b = ");
    double b = Double.parseDouble(System.console().readLine());

    System.out.print("c = ");
    double c = Double.parseDouble(System.console().readLine());


    // 0x^2 + 0x + 0 = 0

    if ((a == 0) && (b == 0) && (c == 0)) {
      System.out.println("La ecuación tiene infinitas soluciones.");
    }
```

```
    // 0x^2 + 0x + c = 0  con c distinto de 0

    if ((a == 0) && (b == 0) && (c != 0)) {
      System.out.println("La ecuación no tiene solución.");
    }

    // ax^2 + bx + 0 = 0  con a y b distintos de 0

    if ((a != 0) && (b != 0) && (c == 0)) {
      System.out.println("x1 = 0");
      System.out.println("x2 = " + (-b / a));
    }


    // 0x^2 + bx + c = 0  con b y c distintos de 0

    if ((a == 0) && (b != 0) && (c != 0)) {
      System.out.println("x1 = x2 = " + (-c / b));
    }

    // ax^2 + bx + c = 0  con a, b y c distintos de 0

    if ((a != 0) && (b != 0) && (c != 0)) {

      double discriminante = b*b - (4 * a * c);

      if (discriminante < 0) {
          System.out.println("La ecuación no tiene soluciones reales");
      }  else {
        System.out.println("x1 = " + (-b + Math.sqrt(discriminante))/(2 * a));
        System.out.println("x2 = " + (-b - Math.sqrt(discriminante))/(2 * a));
      }
    }

  }
}
```

Ejercicio 10

Fichero: S04Ejercicio10.java

```
/**
 * 4. Sentencia Condicional
 *
 * 10. Escribe un programa que nos diga el horóscopo a partir del día y
 *     el mes de nacimiento.
 *
 * @author Luis José Sánchez
 */
public class S04Ejercicio10 {

  public static void main(String[] args) {

    String horoscopo = "";

    System.out.println("Este programa le dirá cuál es su horóscopo a partir de su fecha de nac\
imiento.");
    System.out.print("Introduzca el número del mes en que nació (1-12): ");
    int mes = Integer.parseInt(System.console().readLine());

    System.out.print("Ahora introduzca el día: ");
    int dia = Integer.parseInt(System.console().readLine());

    switch(mes) {
      case 1:
        if (dia < 21) {
          horoscopo = "capricornio";
        } else {
          horoscopo = "acuario";
        }
        break;
      case 2:
        if (dia < 20) {
          horoscopo = "acuario";
        } else {
          horoscopo = "piscis";
        }
        break;
      case 3:
        if (dia < 21) {
          horoscopo = "piscis";
        } else {
          horoscopo = "aries";
        }
        break;
      case 4:
        if (dia < 21) {
```

```
      horoscopo = "aries";
    } else {
      horoscopo = "tauro";
    }
    break;
  case 5:
    if (dia < 20) {
      horoscopo = "tauro";
    } else {
      horoscopo = "géminis";
    }
    break;
  case 6:
    if (dia < 22) {
      horoscopo = "géminis";
    } else {
      horoscopo = "cáncer";
    }
    break;
  case 7:
    if (dia < 22) {
      horoscopo = "cáncer";
    } else {
      horoscopo = "Leo";
    }
    break;
  case 8:
    if (dia < 24) {
      horoscopo = "leo";
    } else {
      horoscopo = "virgo";
    }
    break;
  case 9:
    if (dia < 23) {
      horoscopo = "virgo";
    } else {
      horoscopo = "libra";
    }
    break;
  case 10:
    if (dia < 23) {
      horoscopo = "libra";
    } else {
      horoscopo = "escorpio";
    }
```

```
        break;
      case 11:
        if (dia < 23) {
          horoscopo = "escorpio";
        } else {
          horoscopo = "sagitario";
        }
        break;
      case 12:
        if (dia < 21) {
          horoscopo = "sagitario";
        } else {
          horoscopo = "capricornio";
        }
        break;
      default:
    }

    System.out.print("Su horóscopo es " + horoscopo);

  }
}
```

Ejercicio 11

Fichero: S04Ejercicio11.java

```
/**
 * 4. Sentencia Condicional
 *
 * 11. Escribe un programa que dada una hora determinada (horas y minutos),
 *     calcule los segundos que faltan para llegar a la medianoche.
 *
 * @author Luis José Sánchez
 */
public class S04Ejercicio11 {

  public static void main(String[] args) {

    System.out.print("A continuación deberá introducir una hora del día, ");
    System.out.println("primero introducirá la hora y luego los minutos.");

    System.out.print("hora: ");
    int hora = Integer.parseInt(System.console().readLine());
```

```
    System.out.print("minuto: ");
    int minuto = Integer.parseInt(System.console().readLine());

    int segundosTranscurridos = (hora * 3600) + (minuto * 60);
    int segundosHastaMedianoche = (24 * 3600) - segundosTranscurridos;

    //System.out.println("Desde las " + hora + ":" + minuto + " hasta la medianoche faltan " +\
 segundosHastaMedianoche + " segundos.");
    System.out.printf("Desde las %02d:%02d hasta la medianoche faltan %d segundos", hora, minu\
to, segundosHastaMedianoche);
  }
}
```

Ejercicio 12

Fichero: S04Ejercicio12.java

```
/**
 * 4. Sentencia Condicional
 *
 * 12. Realiza un minicuestionario con 10 preguntas tipo test sobre las
 *     asignaturas que se imparten en el curso. Cada pregunta acertada
 *     sumará un punto. El programa mostrará al final la califcación
 *     obtenida. Pásale el minicuestionario a tus compañeros y pídeles
 *     que lo hagan para ver qué tal andan de conocimientos en las
 *     diferentes asignaturas del curso.
 *
 * @author Luis José Sánchez
 */
public class S04Ejercicio12 {

  public static void main(String[] args) {

    int puntos = 0;
    String respuesta;

    System.out.println("CUESTIONARIO DE 1º DAW");

    System.out.println("1. ¿Cuál de los siguientes tipos de datos de Java tiene más precisión?\
");
    System.out.println("a) int\nb) double\nc) float");
    System.out.print("=> ");
    respuesta = System.console().readLine();
    if (respuesta.equals("b")) {
      puntos++;
```

```
    }

    System.out.println("2. ¿Cuál es el lenguaje que se utiliza para hacer consultas en las bas\
es de datos");
    System.out.println("a) XML\nb) SELECT\nc) SQL");
    System.out.print("=> ");
    respuesta = System.console().readLine();
    if (respuesta.equals("c")) {
      puntos++;
    }

    System.out.println("3. Para insertar un hiperenlace en una página se utiliza la etiqueta..\
.");
    System.out.println("a) href\nb) a\nc) link");
    System.out.print("=> ");
    respuesta = System.console().readLine();
    if (respuesta.equals("b")) {
      puntos++;
    }

    System.out.println("4. ¿En qué directorio se encuentran los archivos de configuración de L\
inux?");
    System.out.println("a) /etc\nb) /config\nc) /cfg");
    System.out.print("=> ");
    respuesta = System.console().readLine();
    if (respuesta.equals("a")) {
      puntos++;
    }

    System.out.println("5. ¿Cuál de las siguientes memorias es volátil?");
    System.out.println("a) RAM\nb) EPROM\nc) ROM");
    System.out.print("=> ");
    respuesta = System.console().readLine();
    if (respuesta.equals("a")) {
      puntos++;
    }

    System.out.println("\nHa obtenido " + puntos + " puntos.");
  }
}
```

Ejercicio 13

Fichero: S04Ejercicio13.java

```java
/**
 * 4. Sentencia Condicional
 *
 * 13. Escribe un programa que ordene tres números enteros introducidos por teclado.
 *
 * @author Luis José Sánchez
 */
public class S04Ejercicio13 {

  public static void main(String[] args) {

    int aux;

    System.out.println("Este programa ordena tres números introducidos por teclado.");
    System.out.println("Por favor, vaya introduciendo los tres números y pulsando INTRO:");
    int a = Integer.parseInt(System.console().readLine());
    int b = Integer.parseInt(System.console().readLine());
    int c = Integer.parseInt(System.console().readLine());

    // ordenación de los dos primeros números
    if (a > b) {
      aux = a;
      a = b;
      b = aux;
    }

    // ordenación de los dos últimos números
    if (b > c) {
      aux = b;
      b = c;
      c = aux;
    }

    // se vuelven a ordenar los dos primeros
    if (a > b) {
      aux = a;
      a = b;
      b = aux;
    }

    System.out.println("Los números introducidos ordenados de menor a mayor son " + a + ", " +\
 b + " y " + c + ".");
  }
}
```

Ejercicio 14

Fichero: S04Ejercicio14.java

```
/**
 * 4. Sentencia Condicional
 *
 * 14. Realiza un programa que diga si un número introducido por teclado
 *     es par y/o divisible entre 5.
 *
 * @author Luis José Sánchez
 */
public class S04Ejercicio14 {

  public static void main(String[] args) {

    System.out.print("Por favor, introduzca un número entero: ");
    int n = Integer.parseInt(System.console().readLine());

    if ((n % 2) == 0) {
      System.out.print("El número introducido es par");
    } else {
      System.out.print("El número introducido es impar");
    }

    if ((n % 5) == 0) {
      System.out.println(" y divisible entre 5.");
    } else {
      System.out.println(" y no es divisible entre 5.");
    }
  }
}
```

Ejercicio 15

Fichero: S04Ejercicio15.java

```java
/**
 * 4. Sentencia Condicional
 *
 * 15. Escribe un programa que pinte una pirámide rellena con un carácter
 *     introducido por teclado que podrá ser una letra, un número o un
 *     símbolo como *, +, -, $, &, etc. El programa debe permitir al
 *     usuario mediante un menú elegir si el vértice de la pirámide está
 *     apuntando hacia arriba, hacia abajo, hacia la izquierda o hacia la
 *     derecha.
 *
 * @author Luis José Sánchez
 */
public class S04Ejercicio15 {

  public static void main(String[] args) {

    System.out.println("Este programa pinta una pirámide.");
    System.out.print("Introduzca un carácter de relleno: ");
    String r = System.console().readLine();
    System.out.println("Elija un tipo de pirámide");
    System.out.println("1. Con el vértice hacia arriba");
    System.out.println("2. Con el vértice hacia abajo");
    System.out.println("3. Con el vértice hacia la izquierda");
    System.out.println("4. Con el vértice hacia la derecha");
    int opcion = Integer.parseInt(System.console().readLine());

    switch(opcion) {
      case 1:
        System.out.println("  " + r);
        System.out.println(" " + r + r + r);
        System.out.println(r + r + r + r + r);
        break;
      case 2:
        System.out.println(r + r + r + r + r);
        System.out.println(" " + r + r + r);
        System.out.println("  " + r);
        break;
      case 3:
        System.out.println("    " + r);
        System.out.println("  " + r + " " + r);
        System.out.println(r + " " + r + " " + r);
        System.out.println("  " + r + " " + r);
        System.out.println("    " + r);
        break;
      case 4:
        System.out.println(r);
```

```
        System.out.println(r + " " + r);
        System.out.println(r + " " + r + " " + r);
        System.out.println(r + " " + r);
        System.out.println(r);
        break;
      default:
    }
  }
}
```

Ejercicio 16

Fichero: S04Ejercicio16.java

```
/**
 * 4. Sentencia Condicional
 *
 * 16. Realiza un programa que nos diga si hay probabilidad de que
 *     nuestra pareja nos está siendo infiel. El programa irá haciendo
 *     preguntas que el usuario contestará con una v (verdadero) o una
 *     f (falso). Cada pregunta contestada con v sumará 3 puntos.
 *     Las preguntas contestadas con f no suman puntos. Utiliza el
 *     fichero test_infidelidad.txt para obtener las preguntas y las
 *     conclusiones del programa.
 *
 * @author Luis José Sánchez
 *
 */
public class S04Ejercicio16 {

  public static void main(String[] args) {

    String respuesta;
    int puntos = 0;

    System.out.println("TEST DE FIDELIDAD");
    System.out.println("Este programa te dirá si hay probabilidad de que tu pareja está siendo\
 infiel.\n");

    System.out.print("1. Tu pareja parece estar más inquieta de lo normal sin ningún motivo ap\
arente.\n(v)erdadero o (f)also: ");
    respuesta = System.console().readLine();
    if ( respuesta.equals("v") ) {
      puntos += 3;
    }
```

```
    System.out.print("2. Ha aumentado sus gastos de vestuario.\n(v)erdadero o (f)also: ");
    respuesta = System.console().readLine();
    if ( respuesta.equals("v") ) {
      puntos += 3;
    }

    System.out.print("3. Ha perdido el interés que mostraba anteriormente por ti.\n(v)erdadero\
 o (f)also: ");
    respuesta = System.console().readLine();
    if ( respuesta.equals("v") ) {
      puntos += 3;
    }

    System.out.print("4. Ahora se afeita y se asea con más frecuencia (si es hombre) o ahora s\
e arregla el pelo y se asea con más frecuencia (si es mujer).\n(v)erdadero o (f)also: ");
    respuesta = System.console().readLine();
    if ( respuesta.equals("v") ) {
      puntos += 3;
    }

    System.out.print("5. No te deja que mires la agenda de su teléfono móvil.\n(v)erdadero o (\
f)also: ");
    respuesta = System.console().readLine();
    if ( respuesta.equals("v") ) {
      puntos += 3;
    }

    System.out.print("6. A veces tiene llamadas que dice no querer contestar cuando estás tú d\
elante.\n(v)erdadero o (f)also: ");
    respuesta = System.console().readLine();
    if ( respuesta.equals("v") ) {
      puntos += 3;
    }

    System.out.print("7. Últimamente se preocupa más en cuidar la línea y/o estar bronceado/a.\
\n(v)erdadero o (f)also: ");
    respuesta = System.console().readLine();
    if ( respuesta.equals("v") ) {
      puntos += 3;
    }

    System.out.print("8. Muchos días viene tarde después de trabajar porque dice tener mucho m\
ás trabajo.\n(v)erdadero o (f)also: ");
    respuesta = System.console().readLine();
    if ( respuesta.equals("v") ) {
```

```
      puntos += 3;
    }

    System.out.print("9. Has notado que últimamente se perfuma más.\n(v)erdadero o (f)also: ");
    respuesta = System.console().readLine();
    if ( respuesta.equals("v") ) {
      puntos += 3;
    }

    System.out.print("10. Se confunde y te dice que ha estado en sitios donde no ha ido contig\
o.\n(v)erdadero o (f)also: ");
    respuesta = System.console().readLine();
    if ( respuesta.equals("v") ) {
      puntos += 3;
    }


    // Muestra el resultado del test

    if ( puntos <= 10 ) {
      System.out.print("¡Enhorabuena! tu pareja parece ser totalmente fiel.");
    }

    if ( (puntos > 11 ) && (puntos <= 22) ) {
      System.out.print("Quizás exista el peligro de otra persona en su vida o en su mente, aun\
que seguramente será algo sin importancia. No bajes la guardia.");
    }

    if ( puntos >= 22 ) {
      System.out.print("Tu pareja tiene todos los ingredientes para estar viviendo un romance \
con otra persona. Te aconsejamos que indagues un poco más y averigües qué es lo que está pasan\
do por su cabeza.");
    }
  }
}
```

Ejercicio 17

Fichero: S04Ejercicio17.java

```java
/**
 * 4. Sentencia Condicional
 *
 * 17. Escribe un programa que diga cuál es la última cifra de un número
 *     entero introducido por teclado.
 *
 * @author Luis José Sánchez
 */
public class S04Ejercicio17 {

  public static void main(String[] args) {

    System.out.print("Por favor, introduzca un número entero: ");
    int n = Integer.parseInt(System.console().readLine());
    System.out.println("La última cifra del número introducido es el " + (n % 10));
  }
}
```

Ejercicio 18

Fichero: S04Ejercicio18.java

```java
/**
 * 4. Sentencia Condicional
 *
 * 18. Escribe un programa que diga cuál es la primera cifra de un número
 *     entero introducido por teclado. Se permiten números de hasta 5 cifras.
 *
 * @author Luis José Sánchez
 */
public class S04Ejercicio18 {

  public static void main(String[] args) {

    int n, primera = 0;

    System.out.print("Por favor, introduzca un número entero (de 5 cifras como máximo): ");
    n = Integer.parseInt(System.console().readLine());

    if ( n < 10 ) {
      primera = n;
    }

    if (( n >= 10 ) && ( n < 100 )) {
      primera = n / 10;
```

```
    }

    if (( n >= 100 ) && ( n < 1000 )) {
      primera = n / 100;
    }

    if (( n >= 1000 ) && ( n < 10000 )) {
      primera = n / 1000;
    }

    if ( n >= 10000 ) {
      primera = n / 10000;
    }

    System.out.println("La primera cifra del número introducido es el " + primera);
  }
}
```

Ejercicio 19

Fichero: S04Ejercicio19.java

```
/**
 * 4. Sentencia Condicional
 *
 * 19. Realiza un programa que nos diga cuántos dígitos tiene un número
 *     entero que puede ser positivo o negativo. Se permiten números de
 *     hasta 5 dígitos.
 *
 * @author Luis José Sánchez
 */
public class S04Ejercicio19 {

  public static void main(String[] args) {

    int n, digitos = 0;

    System.out.print("Por favor, introduzca un número entero (de 5 cifras como máximo): ");
    n = Math.abs(Integer.parseInt(System.console().readLine()));

    if ( n < 10 ) {
      digitos = 1;
    }

    if (( n >= 10 ) && ( n < 100 )) {
```

```
      digitos = 2;
    }

    if (( n >= 100 ) && ( n < 1000 )) {
      digitos = 3;
    }

    if (( n >= 1000 ) && ( n < 10000 )) {
      digitos = 4;
    }

    if ( n >= 10000 ) {
      digitos = 5;
    }

    System.out.println("El número introducido tiene " + digitos + " dígitos.");
  }
}
```

Ejercicio 20

Fichero: S04Ejercicio20.java

```
/**
 * 4. Sentencia Condicional
 *
 * 20. Realiza un programa que diga si un número entero positivo
 *     introducido por teclado es capicúa. Se permiten números de
 *     hasta 5 cifras.
 *
 * @author Luis José Sánchez
 */
public class S04Ejercicio20 {

  public static void main(String[] args) {

    int n;
    boolean capicua = false;

    System.out.print("Por favor, introduzca un número entero (de 5 cifras como máximo): ");
    n = Integer.parseInt(System.console().readLine());


    // número de una cifra
    if (n < 10) {
```

```
      capicua = true;
    }

    // número de dos cifras
    if ((n >= 10) && (n < 100)) {
      if ((n / 10) == (n % 10)) {
        capicua = true;
      }
    }

    // número de tres cifras
    if ((n >= 100) && (n < 1000)) {
      if ((n / 100) == (n % 10)) {
        capicua = true;
      }
    }

    // número de cuatro cifras
    if ((n >= 1000) && (n < 10000)) {
      if (((n / 1000) == (n % 10)) && ((( n / 100 ) % 10)== (( n / 10) % 10))) {
        capicua = true;
      }
    }

    // número de cinco cifras
    if (n >= 10000) {
      if (((n / 10000) == (n % 10) ) && ((((n / 1000) % 10)) == ((n / 10) % 10))) {
        capicua = true;
      }
    }

    if (capicua) {
      System.out.println("El número introducido es capicúa.");
    } else {
      System.out.println("El número introducido no es capicúa.");
    }
  }
}
```

Ejercicio 21

Fichero: S04Ejercicio21.java

```
/**
 * 4. Sentencia Condicional
 *
 * 21. Calcula la nota de un trimestre de la asignatura Programación. El programa
 *     pedirá las dos notas que ha sacado el alumno en los dos primeros controles.
 *     Si la media de los dos controles da un número mayor o igual a 5, el alumno
 *     está aprobado y se mostrará la media. En caso de que la media sea un número
 *     menor que 5, el alumno habrá tenido que hacer el examen de recuperación que
 *     se califica como apto o no apto, por tanto se debe preguntar al usuario
 *     ¿Cuál ha sido el resultado de la recuperación? (apto/no apto). Si el
 *     resultado de la recuperación es apto, la nota será un 5; en caso
 *     contrario, se mantiene la nota media anterior.
 *
 *     Ejemplo 1:
 *     Nota del primer control: 7
 *     Nota del segundo control: 10
 *     Tu nota de Programación es 8.5
 *
 *     Ejemplo 2:
 *     Nota del primer control: 6
 *     Nota del segundo control: 3
 *     ¿Cuál ha sido el resultado de la recuperación? (apto/no apto): apto
 *     Tu nota de Programación es 5
 *
 * @author Luis José Sánchez
 */

public class S04Ejercicio21 {
  public static void main(String[] args) {

    System.out.print("Nota del primer control: ");
    double nota1 = Double.parseDouble(System.console().readLine());

    System.out.print("Nota del segundo control: ");
    double nota2 = Double.parseDouble(System.console().readLine());

    double media = (nota1 + nota2) / 2;

    if (media < 5) {
      System.out.print("¿Cuál ha sido el resultado de la recuperación? (apto/no apto): ");
      String recuperacion = System.console().readLine();
      if (recuperacion.equals("apto")) {
        media = 5;
      }
    }
```

```
    System.out.print("Tu nota de Programación es " + media);
  }
}
```

Ejercicio 22

Fichero: S04Ejercicio22.java

```
/**
 * 4. Sentencia Condicional
 *
 * 22. Realiza un programa que, dado un día de la semana (de lunes a viernes) y
 *     una hora (horas y minutos), calcule cuántos minutos faltan para el fin de
 *     semana. Se considerará que el fin de semana comienza el viernes a las 15:00h.
 *     Se da por hecho que el usuario introducirá un día y hora correctos,
 *     anterior al viernes a las 15:00h.
 *
 * @author Luis José Sánchez
 */

public class S04Ejercicio22 {
  public static void main(String[] args) {

    System.out.print("Por favor, introduzca un día de la semana (de lunes a viernes): ");
    String dia = System.console().readLine();

    int diaNumerico = 0;

    switch(dia) {
      case "lunes":
        diaNumerico = 0;
        break;
      case "martes":
        diaNumerico = 1;
        break;
      case "miércoles":
      case "miercoles":
        diaNumerico = 2;
        break;
      case "jueves":
        diaNumerico = 3;
        break;
      case "viernes":
        diaNumerico = 4;
        break;
```

```
      default:
        System.out.print("El día introducido no es correcto.");
    }

    System.out.println("A continuación introduzca la hora (hora y minutos)");

    System.out.print("Hora: ");
    int horas = Integer.parseInt(System.console().readLine());

    System.out.print("Minutos: ");
    int minutos = Integer.parseInt(System.console().readLine());

    int minutosTotales = (4 * 24 * 60) + (15 * 60);
    int minutosActuales = (diaNumerico * 24 * 60) + (horas * 60) + minutos;

    System.out.print("Faltan " + (minutosTotales - minutosActuales) + " minutos para llegar al\
 fin de semana.");
  }
}
```

Ejercicio 23

Fichero: S04Ejercicio23.java

```
/**
 * Aprende Java con Ejercicios (https://leanpub.com/aprendejava)
 *
 * Capítulo 4. Sentencia condicional.
 *
 * 23. Escribe un programa que calcule el precio final de un producto según su
 *     base imponible (precio antes de impuestos), el tipo de IVA aplicado (general,
 *     reducido o superreducido) y el código promocional. Los tipos de IVA general,
 *     reducido y superreducido son del 21%, 10% y 4% respectivamente. Los códigos
 *     promocionales pueden ser nopro, mitad, meno5 o 5porc que significan
 *     respectivamente que no se aplica promoción, el precio se reduce a la mitad,
 *     se descuentan 5 euros o se descuenta el 5%. El ejercicio se da por bueno
 *     si se muestran los valores correctos, aunque los números no estén tabulados.
 *
 *     Ejemplo:
 *     Introduzca la base imponible: 25
 *     Introduzca el tipo de IVA (general, reducido o superreducido): reducido
 *     Introduzca el código promocional (nopro, mitad, meno5 o 5porc): mitad
 *     Base imponible        25.00
 *     IVA (10%)              2.50
 *     Precio con IVA        27.50
```

```
 *     Cód. promo. (mitad): -13.75
 *     TOTAL                 13.75
 *
 * @author Luis José Sánchez
 */
public class S04Ejercicio23 {
  public static void main(String[] args) {

    System.out.print("Introduzca la base imponible: ");
    double baseImponible = Double.parseDouble(System.console().readLine());

    System.out.print("Introduzca el tipo de IVA (general, reducido o superreducido): ");
    String tipoIVA = System.console().readLine();

    System.out.print("Introduzca el código promocional (nopro, mitad, meno5 o 5porc): ");
    String codigoPromocional = System.console().readLine();

    // Calcula el IVA y el precio antes del descuento

    int tipoIVANumerico = 0;

    switch(tipoIVA) {
      case "general":
        tipoIVANumerico = 21;
        break;
      case "reducido":
        tipoIVANumerico = 10;
        break;
      case "superreducido":
        tipoIVANumerico = 4;
        break;
      default:
        System.out.println("El tipo de IVA introducido no es correcto.");
    }

    double iva = baseImponible * tipoIVANumerico / 100;
    double precioSinDescuento = baseImponible + iva;

    // Calcula el descuento

    double descuento = 0;

    switch(codigoPromocional) {
      case "nopro":
        break;
      case "mitad": // el precio se reduce a la mitad
```

```
        descuento = precioSinDescuento / 2;
        break;
      case "meno5": // se descuentan 5 euros
        descuento = 5;
        break;
      case "5porc": // se descuenta el 5%
        descuento = precioSinDescuento * 0.05;
        break;
      default:
        System.out.println("El código promocional introducido no es correcto.");
    }

    // Muestra el precio final del producto desglosado

    double total = precioSinDescuento - descuento;

    System.out.printf("Base imponible       %6.2f\n", baseImponible);
    System.out.printf("IVA (%2d%%)            %6.2f\n", tipoIVANumerico, iva);
    System.out.printf("Precio con IVA       %6.2f\n", precioSinDescuento);
    System.out.printf("Cód. promo. (%5s) -%6.2f\n", codigoPromocional, descuento);
    System.out.printf("TOTAL                %6.2f", total);
  }
}
```

Ejercicio 24

Fichero: S04Ejercicio24.java

```
/**
 * Aprende Java con Ejercicios
 * https://leanpub.com/aprendejava
 *
 * Capítulo 4. Sentencia condicional.
 *
 * Ejercicio 24
 *
 * @author Luis José Sánchez
 */
public class S04Ejercicio24 {
  public static void main(String[] args) {

    System.out.println("1 - Programador junior");
    System.out.println("2 - Prog. senior");
    System.out.println("3 - Jefe de proyecto");
    System.out.print("Introduzca el cargo del empleado (1 - 3): ");
```

```
int cargo = Integer.parseInt(System.console().readLine());

System.out.print("¿Cuántos días ha estado de viaje visitando clientes? ");
int diasVisita = Integer.parseInt(System.console().readLine());

System.out.print("Introduzca su estado civil (1 - Soltero, 2 - Casado): ");
int estadoCivil = Integer.parseInt(System.console().readLine());

double sueldoBase = 0;

switch(cargo) {
  case 1: // Programador junior
    sueldoBase = 950;
    break;
  case 2: // Programador senior
    sueldoBase = 1200;
    break;
  case 3: // jefe de proyecto
    sueldoBase = 1600;
    break;
  default:
    System.out.println("No ha elegido correctamente el sueldo base.");
}

double sueldoDietas = diasVisita * 30;

double sueldoBruto = sueldoBase + sueldoDietas;

double irpf = 0;

if (estadoCivil == 1) { // Soltero
  irpf = 25;
} else if (estadoCivil == 2) { // Casado
  irpf = 20;
} else {
  System.out.println("No ha elegido correctamente el estado civil.");
}

double cuantiaIrpf = (sueldoBruto * irpf) / 100;

// Muestra la nómina desglosada
System.out.println("□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□");
System.out.printf("□ Sueldo base            %7.2f □\n", sueldoBase);
System.out.printf("□ Dietas (%2d viajes)     %7.2f □\n", diasVisita, sueldoDietas);
System.out.println("□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□");
System.out.printf("□ Sueldo bruto           %7.2f □\n", sueldoBruto);
```

```
    System.out.printf("□ Retención IRPF (%.0f%%)   %7.2f □\n", irpf, cuantiaIrpf);
    System.out.println("□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□");
    System.out.printf("□ Sueldo neto            %7.2f □\n", sueldoBruto - cuantiaIrpf);
    System.out.println("□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□");
  }
}
```

Ejercicio 25

Fichero: S04Ejercicio25.java

```
import java.util.Scanner;

/**
 * Aprende Java con Ejercicios
 * https://leanpub.com/aprendejava
 *
 * Capítulo 4. Sentencia condicional.
 *
 * Ejercicio 25
 *
 * @author Luis José Sánchez
 */
public class S04Ejercicio25 {

  public static void main(String[] args) {

    Scanner s = new Scanner(System.in);
    System.out.print("Introduzca la altura de la bandera en cm: ");
    int altura = Integer.parseInt(s.nextLine());
    System.out.print("Ahora introduzca la anchura: ");
    int anchura = Integer.parseInt(s.nextLine());
    System.out.print("¿Quiere escudo bordado? (s/n): ");
    boolean conEscudo = ((s.nextLine()).toLowerCase()).equals("s");

    String escudoCadena;
    float precioEscudo;

    if (conEscudo) {
      escudoCadena = "Con escudo";
      precioEscudo = 2.50f;
    } else {
      escudoCadena = "Sin escudo";
      precioEscudo = 0;
    }
```

```
    System.out.println("Gracias. Aquí tiene el desglose de su compra.");
    System.out.printf("Bandera de %5d cm2: %5.2f €\n", anchura * altura, (float)anchura * altu\
ra / 100);
    System.out.printf("%s:           %5.2f €\n", escudoCadena, precioEscudo);
    System.out.printf("Gastos de envío:      %5.2f €\n", 3.25);
    System.out.printf("Total:                %5.2f €\n", (float)anchura * altura / 100 + preci\
oEscudo + 3.25);
  }
}
```

Ejercicio 26

Fichero: S04Ejercicio26.java

```
import java.util.Scanner;

public class S04Ejercicio26 {
  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);

    System.out.println("Venta de entradas CineCampa");

    System.out.print("Número de entradas: ");
    int entradas = Integer.parseInt(s.nextLine());

    System.out.print("Día de la semana: ");
    String dia = s.nextLine();

    System.out.print("¿Tiene tarjeta CineCampa? (s/n): ");
    boolean tieneTarjeta = (s.nextLine()).equals("s");

    int entradasIndividuales = entradas;
    double precioEntradaIndividual = 8;
    int entradasDePareja = 0;
    double total = 0;
    double descuento = 0;
    double aPagar = 0;

    switch (dia) {
      case "miércoles":
      case "miercoles":
        precioEntradaIndividual = 5;
      case "jueves":
        entradasDePareja = entradas / 2;
```

```
        entradasIndividuales = entradas % 2;
      default:
    }

    total = precioEntradaIndividual * entradasIndividuales;
    total += 11 * entradasDePareja;

    if(tieneTarjeta) {
      descuento = total * 0.1;
    }

    aPagar = total - descuento;

    System.out.println("\nAquí tiene sus entradas. Gracias por su compra.");

    if (entradasDePareja > 0) {
      System.out.printf("Entradas de pareja            %2d\n", entradasDePareja);
      System.out.printf("Precio por entrada de pareja  %5.2f €\n", 11.0);
    }

    if (entradasIndividuales > 0) {
      System.out.printf("Entradas individuales         %2d\n", entradasIndividuales);
      System.out.printf("Precio por entrada individual %5.2f €\n", precioEntradaIndividual);
    }

    System.out.printf("Total                         %5.2f €\n", total);
    System.out.printf("Descuento                     %5.2f €\n", descuento);
    System.out.printf("A pagar                       %5.2f €\n", aPagar);
  }
}
```

Ejercicio 27

Fichero: S04Ejercicio27.java

```
import java.util.Scanner;

public class S04Ejercicio27 {

  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);

    System.out.print("Elija un sabor (manzana, fresa o chocolate): ");
    String sabor = s.nextLine();
```

```
double precioSabor = 0;
double precioNata = 0;
double precioNombre = 0;
String tipoChocolate = "";

switch (sabor) {
  case "manzana":
    precioSabor = 18;
    break;
  case "fresa":
    precioSabor = 16;
    break;
  case "chocolate":
    System.out.print("¿Qué tipo de chocolate quiere? (negro o blanco): ");
    tipoChocolate = s.nextLine();
    if (tipoChocolate.equals("negro")) {
      precioSabor = 14;
    } else if (tipoChocolate.equals("blanco")) {
      precioSabor = 15;
    }
    break;
  default:
}

System.out.print("¿Quiere nata? (si o no): ");
boolean conNata = s.nextLine().equals("si");

System.out.print("¿Quiere ponerle un nombre? (si o no): ");
boolean conNombre = s.nextLine().equals("si");

System.out.print("Tarta de " + sabor);

if (sabor.equals("chocolate")) {
  System.out.print(" " + tipoChocolate);
}

System.out.printf(": %.2f €\n", precioSabor);

if (conNata) {
  precioNata = 2.5;
  System.out.printf("Con nata: %.2f €\n", precioNata);
}

if (conNombre) {
  precioNombre = 2.75;
  System.out.printf("Con nombre: %.2f €\n", precioNombre);
```

```
    }

    System.out.printf("Total: %.2f €\n", precioSabor + precioNata + precioNombre);
  }

}
```

Ejercicio 28

Fichero: S04Ejercicio28.java

```
import java.util.Scanner;

public class S04Ejercicio28 {

  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);

    System.out.print("Turno del jugador 1 (introduzca piedra, papel o tijera): ");
    String jugada1 = s.nextLine();

    System.out.print("Turno del jugador 1 (introduzca piedra, papel o tijera): ");
    String jugada2 = s.nextLine();

    if (jugada1.equals(jugada2)) {
      System.out.println("Empate");
    } else {
      int ganador = 2;
      switch(jugada1) {
        case "piedra":
          if (jugada2.equals("tijera")) {
            ganador = 1;
          }
          break;
        case "papel":
          if (jugada2.equals("piedra")) {
            ganador = 1;
          }
          break;
        case "tijera":
          if (jugada2.equals("papel")) {
            ganador = 1;
          }
          break;
        default:
```

```
      }
      System.out.println("Gana el jugador " + ganador);
    }
  }

}
```

Ejercicio 29

Fichero: S04Ejercicio29.java

```
import java.util.Scanner;

public class S04Ejercicio29 {

  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);

    String pitufo = "";
    String resultado = "";
    double precioComida = 0;
    double precioBebida = 0;

    // Comida
    System.out.print("¿Qué ha tomado de comer? (palmera, donut o pitufo): ");
    String comida = s.nextLine();

    if (comida.equalsIgnoreCase("pitufo")) {
      System.out.print("¿Con qué se ha tomado el pitufo? (aceite o tortilla): ");
      pitufo = s.nextLine();

      if (pitufo.equalsIgnoreCase("aceite")) {
        resultado = "Pitufo con aceite: 1,20 €";
        precioComida = 1.20;
      } else if (pitufo.equalsIgnoreCase("tortilla")) {
        resultado = "Pitufo con tortilla: 1,60 €";
        precioComida = 1.60;
      }
    } else if (comida.equalsIgnoreCase("palmera")) {
      resultado = "Palmera: 1,40 €";
      precioComida = 1.40;
    } else if (comida.equalsIgnoreCase("donut")) {
      resultado = "Donut: 1,00 €";
      precioComida = 1.00;
    }
```

```
    // Bebida
    System.out.print("¿Qué ha tomado de beber? (zumo o café): ");
    String bebida = s.nextLine();
    if (bebida.equalsIgnoreCase("zumo")) {
      resultado += "\nZumo: 1,50 €";
      precioBebida = 1.50;
    } else if (bebida.equalsIgnoreCase("café")) {
      resultado += "\nCafé: 1,20 €";
      precioBebida = 1.20;
    }

    System.out.println(resultado);
    System.out.printf("Total desayuno: %.2f €\n", precioComida + precioBebida);
  }

}
```

Bucles

Ejercicio 1

Fichero: S05Ejercicio01.java

```
/**
 * 5. Bucles
 *
 * 1. Muestra los números múltiplos de 5 de 0 a 100 utilizando un bucle for.
 *
 * @author Luis José Sánchez
 */
public class S05Ejercicio01 {

  public static void main(String[] args) {

    for(int i = 0; i <= 100; i += 5) {
      System.out.println(i);
    }
  }
}
```

Ejercicio 2

Fichero: S05Ejercicio02.java

```
/**
 * 5. Bucles
 *
 * 2. Muestra los números múltiplos de 5 de 0 a 100 utilizando un bucle while.
 *
 * @author Luis José Sánchez
 */
public class S05Ejercicio02 {

  public static void main(String[] args) {

    int i = 0;

    while(i <= 100) {
      System.out.println(i);
      i+=5;
    }
```

```
  }
}
```

Ejercicio 3

Fichero: S05Ejercicio03.java

```
/**
 * 5. Bucles
 *
 * 3. Muestra los números múltiplos de 5 de 0 a 100 utilizando un bucle do-while.
 *
 * @author Luis José Sánchez
 */

public class S05Ejercicio03 {

  public static void main(String[] args) {

    int i = 0;

    do {
      System.out.println(i);
      i+=5;
    } while(i <= 100);
  }
}
```

Ejercicio 4

Fichero: S05Ejercicio04.java

```
/**
 * 5. Bucles
 *
 * 4. Muestra los números del 320 al 160, contando de 20 en 20 hacia
 *    atrás utilizando un bucle for.
 *
 * @author Luis José Sánchez
 */

public class S05Ejercicio04 {

  public static void main(String[] args) {
```

```
    for(int i = 320; i >= 160; i-=20) {
      System.out.println(i);
    }
  }
}
```

Ejercicio 5

Fichero: S05Ejercicio05.java

```
/**
 * 5. Bucles
 *
 * 5. Muestra los números del 320 al 160, contando de 20 en 20 hacia
 *    atrás utilizando un bucle while.
 *
 * @author Luis José Sánchez
 */
public class S05Ejercicio05 {

  public static void main(String[] args) {

    int i = 320;

    while(i >= 160) {
        System.out.println(i);
        i-=20;
    }
  }
}
```

Ejercicio 6

Fichero: S05Ejercicio06.java

```java
/**
 * 5. Bucles
 *
 * 6. Muestra los números del 320 al 160, contando de 20 en 20
 *    utilizando un bucle do-while.
 *
 * @author Luis José Sánchez
 */
public class S05Ejercicio06 {

  public static void main(String[] args) {

    int i = 320;

    do {
      System.out.println(i);
      i-=20;
    } while(i >= 160);
  }
}
```

Ejercicio 7

Fichero: S05Ejercicio07.java

```java
/**
 * 5. Bucles
 *
 * 7. Realiza el control de acceso a una caja fuerte. La combinación
 *    será un número de 4 cifras. El programa nos pedirá la combinación
 *    para abrirla. Si no acertamos, se nos mostrará el mensaje
 *    “Lo siento, esa no es la combinación” y si acertamos se nos dirá
 *    “La caja fuerte se ha abierto satisfactoriamente”. Tendremos cuatro
 *    oportunidades para abrir la caja fuerte.
 *
 * @author Luis José Sánchez
 */
public class S05Ejercicio07 {

  public static void main(String[] args) {

    int intentos = 4;
    int numeroIntroducido;
    boolean acertado = false;
```

```
    do {
      System.out.print("Introduzca la clave de la caja fuerte: ");
      numeroIntroducido = Integer.parseInt(System.console().readLine());

      if (numeroIntroducido == 8888) {
        acertado = true;
      } else {
        System.out.println("Clave incorrecta");
      }

      intentos--;

    } while((intentos > 0) && (!acertado));

    if (acertado) {
      System.out.println("Ha abierto la caja fuerte.");
    } else {
      System.out.println("Lo siento, ha agotado las 4 oportunidades.");
    }
  }
}
```

Ejercicio 8

Fichero: S05Ejercicio08.java

```
/**
 * 5. Bucles
 *
 * 8. Muestra la tabla de multiplicar de un número introducido por teclado.
 *
 * @author Luis José Sánchez
 */
public class S05Ejercicio08 {

  public static void main(String[] args) {

    System.out.print("Introduzca un número entero y le mostraré la tabla de multiplicar: ");
    int numeroIntroducido = Integer.parseInt(System.console().readLine());

    for (int i = 0; i <= 10; i++) {
      System.out.println(numeroIntroducido + " x " + i + " = " + numeroIntroducido * i);
    }

  }
}
```

Ejercicio 9

Fichero: S05Ejercicio09.java

```
/**
 * 5. Bucles
 *
 * 9. Realiza un programa que nos diga cuántos dígitos tiene un número introducido por teclado.
 *
 * @author Luis José Sánchez
 */

public class S05Ejercicio09 {

  public static void main(String[] args) {

    int numeroDeDigitos = 1, n, numeroIntroducido;

    System.out.print("Introduzca un número entero y le diré cuántos dígitos tiene: ");
    numeroIntroducido = Integer.parseInt(System.console().readLine());

    n = numeroIntroducido;

    while (n > 10) {
      n /= 10;
      numeroDeDigitos++;
    }

    System.out.println(numeroIntroducido + " tiene " + numeroDeDigitos + " dígito/s.");

  }
}
```

Ejercicio 10

Fichero: S05Ejercicio10.java

```
/**
 * 5. Bucles
 *
 * 10. Escribe un programa que calcule la media de un conjunto de números
 *     positivos introducidos por teclado. A priori, el programa no sabe
 *     cuántos números se introducirán. El usuario indicará que ha terminado
 *     de introducir los datos cuando meta un número negativo.
 *
 * @author Luis José Sánchez
 */

public class S05Ejercicio10 {

  public static void main(String[] args) {

    double numeros = 0;
    double numeroIntroducido = 0;
    double suma = 0;

    System.out.println("Este programa calcula la media de los números positivos introducidos."\
);
    System.out.println("Vaya introduciendo números (puede parar introduciendo un número negati\
vo):");

    while (numeroIntroducido >= 0) {
      numeroIntroducido = Double.parseDouble(System.console().readLine());
      numeros++;
      suma += numeroIntroducido;
    }

    System.out.println("La media de los números positivos introducidos es " + (suma - numeroIn\
troducido)/ (numeros - 1));

  }
}
```

Ejercicio 11

Fichero: S05Ejercicio11.java

```
/**
 * 5. Bucles
 *
 * 11. Escribe un programa que muestre en tres columnas, el cuadrado y
 *     el cubo de los 5 primeros números enteros a partir de uno que se
 *     introduce por teclado.
 *
 * @author Luis José Sánchez
 */

public class S05Ejercicio11 {

  public static void main(String[] args) {

    System.out.print("Introduzca un número: ");
    int numeroIntroducido = Integer.parseInt(System.console().readLine());

    for (int i = numeroIntroducido; i < numeroIntroducido + 5; i++) {
      System.out.printf("%4d %6d %8d\n", i, i * i, i * i * i);
    }
  }
}
```

Ejercicio 12

Fichero: S05Ejercicio12.java

```
/**
 * 5. Bucles
 *
 * 12. Escribe un programa que muestre los n primeros términos de la
 *     serie de Fibonacci. El primer término de la serie de Fibonacci
 *     es 0, el segundo es 1 y el resto se calcula sumando los dos
 *     anteriores, por lo que tendríamos que los términos son 0, 1, 1,
 *     2, 3, 5, 8, 13, 21, 34, 55, 89, 144... El número n se debe
 *     introducir por teclado.
 *
 * @author Luis José Sánchez
 */

public class S05Ejercicio12 {

  public static void main(String[] args) {

    System.out.println("Este programa muestra los n primeros números de la serie de Fibonacci.\
```

```
");
    System.out.print("Por favor, introduzca n: ");
    int n = Integer.parseInt(System.console().readLine());

    switch (n) {
      case 1:
        System.out.print("0");
        break;
      case 2:
        System.out.print("0 1");
        break;
      default:
        System.out.print("0 1");
        int f1 = 0;
        int f2 = 1;
        int aux;
        while(n > 2) {
            aux = f1;
            f1 = f2;
            f2 = aux + f2;
            System.out.print(" " + f2);
            n--;
        }
    }
    System.out.println();
  }
}
```

Ejercicio 13

Fichero: S05Ejercicio13.java

```
/**
 * 5. Bucles
 *
 * 13. Escribe un programa que lea una lista de diez números y determine
 *     cuántos son positivos, y cuántos son negativos.
 *
 * @author Luis José Sánchez
 */

public class S05Ejercicio13 {

  public static void main(String[] args) {
```

```
    System.out.println("Por favor, introduzca 10 números enteros: ");

    int negativos = 0;
    int positivos = 0;

    for (int i = 0; i < 10; i++) {
      if (Integer.parseInt(System.console().readLine()) < 0) {
        negativos++;
      } else {
        positivos++;
      }
    }

    System.out.println("Ha introducido " + positivos + " positivos y " + negativos + " negativ\
os");

  }
}
```

Ejercicio 14

Fichero: S05Ejercicio14.java

```
/**
 * 5. Bucles
 *
 * 14. Escribe un programa que pida una base y un exponente (entero positivo)
 *     y que calcule la potencia.
 *
 * @author Luis José Sánchez
 */

public class S05Ejercicio14 {

  public static void main(String[] args) {

    System.out.println("Cálculo de una potencia");

    System.out.print("Introduzca la base: ");
    int base = Integer.parseInt(System.console().readLine());

    System.out.print("Introduzca el exponente: ");
    int exponente = Integer.parseInt(System.console().readLine());

    double potencia = 1;
```

```
    if (exponente == 0) {
      potencia = 1;
    }

    if (exponente > 0) {
      for (int i = 0; i < exponente; i++) {
        potencia *= base;
      }
    }

    if (exponente < 0) {
      for (int i = 0; i < -exponente; i++) {
        potencia *= base;
      }

      potencia = 1/potencia;
    }

    System.out.println(base + "^" + exponente + " = " + potencia);
  }
}
```

Ejercicio 15

Fichero: S05Ejercicio15.java

```
/**
 * 5. Bucles
 *
 * 15. Escribe un programa que dados dos números, uno real (base) y un
 *     entero positivo (exponente), saque por pantalla todas las potencias
 *     con base el numero dado y exponentes entre uno y el exponente introducido.
 *     No se deben utilizar funciones de exponenciación. Por ejemplo, si
 *     introducimos el 2 y el 5, se deberán mostrar 2¹, 2², 2³, 2□ y 2□.
 *
 * @author Luis José Sánchez
 */

public class S05Ejercicio15 {

  public static void main(String[] args) {

    System.out.print("Introduzca un número real como base: ");
    double base = Double.parseDouble(System.console().readLine());
```

```
    System.out.print("Introduzca un múmero entero como exponente: ");
    int exponenteFinal = Integer.parseInt(System.console().readLine());

    double potencia;
    int exponente;

    for (int i = 1; i <= exponenteFinal; i++) {

      potencia = 1;
      exponente = i;

      for (int j = 0; j < exponente; j++) {
        potencia *= base;
      }

      System.out.println(base + "^" + i + " = " + potencia);
    }

  }
}
```

Ejercicio 16

Fichero: S05Ejercicio16.java

```
/**
 * 5. Bucles
 *
 * 16. Escribe un programa que diga si un número introducido por teclado
 *     es o no primo. Un número primo es aquel que sólo es divisible entre
 *     él mismo y la unidad.
 *
 * @author Luis José Sánchez
 */

public class S05Ejercicio16 {

  public static void main(String[] args) {

    System.out.print("Introduzca un número entero y le diré si es primo: ");
    int numeroIntroducido = Integer.parseInt(System.console().readLine());

    boolean esPrimo = true;
```

```
    for (int i = 2; i < numeroIntroducido; i++) {
      if ((numeroIntroducido % i) == 0) {
        esPrimo = false;
      }
    }

    if (esPrimo) {
      System.out.println("El número introducido es primo.");
    } else {
      System.out.println("El número introducido no es primo.");
    }

  }
}
```

Ejercicio 17

Fichero: S05Ejercicio17.java

```
/**
 * 5. Bucles
 *
 * 17. Realiza un programa que sume los 100 números  siguientes a un número entero y positivo
 *     introducido por teclado. Se debe comprobar que el dato introducido es correcto (que es
 *     un número positivo).
 *
 * @author Luis José Sánchez
 */
public class S05Ejercicio17 {

  public static void main(String[] args) {

    int numeroIntroducido = 0;

    do {
      System.out.print("Introduzca un número entero positivo: ");
      numeroIntroducido = Integer.parseInt(System.console().readLine());

      if(numeroIntroducido < 0) {
        System.out.println("El número introducido no es correcto, debe introducir un número po\
sitivo.");
      }
    } while (numeroIntroducido < 0);

    int suma = 0;
```

```
    for(int i = numeroIntroducido; i < numeroIntroducido + 100; i++) {
      suma += i;
    }

    System.out.println("La suma de los 100 números siguientes a " + numeroIntroducido + " es "\
 + suma);
  }
}
```

Ejercicio 18

Fichero: S05Ejercicio18.java

```
/**
 * 5. Bucles
 *
 * 18. Escribe un programa que obtenga los números enteros comprendidos entre dos números
 *     introducidos por teclado y validados como distintos, el programa debe empezar por
 *     el menor de los enteros introducidos e ir incrementando de 7 en 7.
 *
 * @author Luis José Sánchez
 */

public class S05Ejercicio18 {

  public static void main(String[] args) {

    int primerNumero;
    int segundoNumero;

    // pide dos números y se verifica que sean distintos
    do {
      System.out.print("Introduzca un número entero: ");
      primerNumero = Integer.parseInt(System.console().readLine());
      System.out.print("Introduzca otro número entero distinto al anterior: ");
      segundoNumero = Integer.parseInt(System.console().readLine());

      if(primerNumero == segundoNumero) {
        System.out.println("Los números introducidos no son válidos, deben ser distintos.");
      }
    } while (primerNumero == segundoNumero);

    // si el primer número es mayor que el segundo, se intercambian los valores
    if (primerNumero > segundoNumero) {
```

```
      int aux = primerNumero;
      primerNumero = segundoNumero;
      segundoNumero = aux;
    }

    for(int i = primerNumero; i <= segundoNumero; i += 7) {
      System.out.print(i + " ");
    }

    System.out.println();
  }
}
```

Ejercicio 19

Fichero: S05Ejercicio19.java

```
/**
 * 5. Bucles
 *
 * 19. Realiza un programa que pinte una pirámide por pantalla. La altura
 *     se debe pedir por teclado. El carácter con el que se pinta la pirámide
 *     también se debe pedir por teclado.
 *
 * @author Luis José Sánchez
 */
public class S05Ejercicio19 {

  public static void main(String[] args) {

    System.out.print("Por favor, introduzca la altura de la pirámide: ");
    int alturaIntroducida = Integer.parseInt(System.console().readLine());

    System.out.print("Introduzca el carácter de relleno: ");
    String relleno = System.console().readLine();

    int planta = 1;
    int longitudDeLinea = 1;
    int espacios = alturaIntroducida-1;

    while (planta <= alturaIntroducida) {

      // inserta espacios
      for (int i = 1; i <= espacios; i++) {
        System.out.print(" ");
```

```
      }

      // pinta la línea
      for (int i = 1; i <= longitudDeLinea; i++) {
        System.out.print(relleno);
      }

      System.out.println();

      planta++;
      espacios--;
      longitudDeLinea += 2;
    }
  }
}
```

Ejercicio 20

Fichero: S05Ejercicio20.java

```
/**
 * 5. Bucles
 *
 * 20. Igual que el ejercicio anterior pero esta vez se debe pintar una
 *     pirámide hueca.
 *
 * @author Luis José Sánchez
 */
public class S05Ejercicio20 {

  public static void main(String[] args) {

    System.out.print("Por favor, introduzca la altura de la pirámide: ");
    int alturaIntroducida = Integer.parseInt(System.console().readLine());

    System.out.print("Introduzca el carácter de relleno: ");
    String relleno = System.console().readLine();

    int altura = 1;
    int i = 0;
    int espaciosPorDelante = alturaIntroducida - 1;
    int espaciosInternos = 0;

    while (altura < alturaIntroducida) {
```

```
      // inserta espacios delante
      for (i = 1; i <= espaciosPorDelante; i++) {
        System.out.print(" ");
      }

      // pinta la línea
      System.out.print(relleno);
      for (i = 1; i < espaciosInternos; i++) {
        System.out.print(" ");
      }

      if (altura>1) {
        System.out.print(relleno);
      }

      System.out.println();
      altura++;
      espaciosPorDelante--;
      espaciosInternos += 2;
    } // while ////////////////////////////

    // base de la pirámide
    for (i = 1; i < altura*2; i++) {
      System.out.print(relleno);
    }
  }
}
```

Ejercicio 21

Fichero: S05Ejercicio21.java

```
/**
 * 5. Bucles
 *
 * 21. Realiza un programa que vaya pidiendo números hasta que se introduzca un numero negativ\
o y
 *     nos diga cuantos números se han introducido, la media de los impares y el mayor de los p\
ares.
 *     El número negativo sólo se utiliza para indicar el final de la introducción de datos per\
o no
 *     se incluye en el cómputo.
 *
 * @author Luis José Sánchez
 */
```

```
public class S05Ejercicio21 {

  public static void main(String[] args) {

    System.out.println("Por favor, vaya introduciendo números enteros.");
    System.out.println("Puede terminar mediante la introducción de un número negativo.");

    int numeroIntroducido;
    int numeroDeElementos = 0;
    int sumaImpares = 0;
    int numeroDeElementosImpares = 0;
    int maximoPar = 0;

    do {
      numeroIntroducido = Integer.parseInt(System.console().readLine());

      if (numeroIntroducido >= 0) {
        numeroDeElementos++;
        if ((numeroIntroducido % 2) == 1) { // número impar
          sumaImpares += numeroIntroducido;
          numeroDeElementosImpares++;
        } else { // número par
          if (numeroIntroducido > maximoPar)
            maximoPar = numeroIntroducido;
        }
      }
    } while (numeroIntroducido >= 0);

    System.out.println("Ha introducido " + numeroDeElementos + " números");
    System.out.println("La media de los impares es " + sumaImpares/numeroDeElementosImpares);
    System.out.println("El máximo de los pares es " + maximoPar);
  }
}
```

Ejercicio 22

Fichero: S05Ejercicio22.java

```
/**
 * Aprende Java con Ejercicios (https://leanpub.com/aprendejava)
 *
 * Capítulo 5. Bucles.
 *
 * Ejercicio 22
 *
 * @author Luis José Sánchez
 */
public class S05Ejercicio22 {

  public static void main(String[] args) {

    System.out.print("Números primos entre 2 y 100: ");

    boolean esPrimo = true;

    for (int n = 2; n <= 100; n++) {

      // comprueba si n es primo
      esPrimo = true;
      for (int i = 2; i < n; i++) {
        if (n % i == 0) {
          esPrimo = false;
        }
      }

      // si n es primo, se muestra por pantalla
      if (esPrimo) {
        System.out.print(n + " ");
      }
    }
    System.out.println();
  }
}
```

Ejercicio 23

Fichero: S05Ejercicio23.java

```
/**
 * 5. Bucles
 *
 * 23. Escribe un programa que permita ir introduciendo una serie indeterminada
 *     de números mientras su suma no supere  el valor 10000. Cuando esto último
 *     ocurra, se debe mostrar el total acumulado, el contador de los números
 *     introducidos y la media.
 *
 * @author Luis José Sánchez
 */
public class S05Ejercicio23 {

  public static void main(String[] args) {

    System.out.println("Por favor, vaya introduciendo números.");
    System.out.println("El programa terminará cuando la suma de los números sea mayor que 1000\
0.");

    int numeroIntroducido;
    int suma = 0;
    int numeroDeElementos = 0;

    do {
      numeroIntroducido = Integer.parseInt(System.console().readLine());
      suma += numeroIntroducido;
      numeroDeElementos++;
    } while (suma <= 10000);

    System.out.println("Ha introducido un total de " + numeroDeElementos + " números.");
    System.out.println("La suma total es " + suma + ".");
    System.out.println("La media es " + suma / numeroDeElementos + ".");
  }
}
```

Ejercicio 24

Fichero: S05Ejercicio24.java

```java
/**
 * 5. Bucles
 *
 * 24. Escribe un programa que lea un número N e imprima una pirámide de números con N filas c\
omo
 *     en la siguiente figura:
 *          1
 *         121
 *        12321
 *       1234321
 *
 * @author Luis José Sánchez
 */
public class S05Ejercicio24 {

  public static void main(String[] args) {

    System.out.println("Este programa pinta una pirámide hecha a base de números.");
    System.out.print("Por favor, introduzca la altura de la pirámide:");
    int alturaIntroducida = Integer.parseInt(System.console().readLine());

    int altura = 1;
    int i = 0;
    int espacios = alturaIntroducida - 1;

    while (altura <= alturaIntroducida) {

      // inserta espacios
      for (i = 1; i <= espacios; i++) {
        System.out.print(" ");
      }

      // pinta la línea de números
      for (i = 1; i < altura; i++) {
        System.out.print(i);
      }

      for (i = altura; i > 0; i--) {
        System.out.print(i);
      }

      System.out.println();

      altura++;
      espacios--;
    } // while
```

```
  }
}
```

Ejercicio 25

Fichero: S05Ejercicio25.java

```
/**
 * 5. Bucles
 *
 * 25. Realiza un programa que pida un número por teclado y que luego
 *     muestre ese número al revés.
 *
 * @author Luis José Sánchez
 */
public class S05Ejercicio25 {

  public static void main(String[] args) {

    System.out.print("Introduzca un número entero: ");
    int numeroIntroducido = Integer.parseInt(System.console().readLine());

    int numero = numeroIntroducido;
    int volteado = 0;

    while (numero > 0) {
      volteado = (volteado * 10) + (numero % 10);
      numero /= 10;
    } // while

    System.out.println("Si le damos la vuelta al " + numeroIntroducido + " tenemos el " + volt\
eado);
  }
}
```

Ejercicio 26

Fichero: S05Ejercicio26.java

```
/**
 * 5. Bucles
 *
 * 26. Realiza un programa que pida primero un número y a continuación un dígito. El programa \
nos
 *     debe dar la posición (o posiciones) contando de izquierda a derecha que ocupa ese dígit\
o en
 *     el número introducido.
 *
 * @author Luis José Sánchez
 */
public class S05Ejercicio26 {

  public static void main(String[] args) {

    System.out.print("Introduzca un número entero: ");
    int numeroIntroducido = Integer.parseInt(System.console().readLine());

    System.out.print("Introduzca un dígito: ");
    int digito = Integer.parseInt(System.console().readLine());

    System.out.print("Contando de izquierda a derecha, el " + digito + " aparece dentro de " +\
 numeroIntroducido + " en las siguientes posiciones: ");

    // le da la vueta al número y calcula la longitud
    int numero = numeroIntroducido;
    int volteado = 0;
    int longitud = 0;
    int posicion = 1;

    if (numero == 0) {
      longitud = 1;
    }

    while (numero > 0) {
      volteado = (volteado * 10) + (numero % 10);
      numero /= 10;
      longitud++;
    } // while

    // comprueba la posición
    while (volteado > 0) {
      if ((volteado % 10) == digito) {
        System.out.print(posicion + " ");
      }
      volteado /= 10;
```

```
      posicion++;
    } // while

    System.out.println();
  }
}
```

Ejercicio 27

Fichero: S05Ejercicio27.java

```
/**
 * 5. Bucles
 *
 * 27. Escribe un programa que muestre, cuente y sume los  múltiplos de 3 que hay entre 1 y un
 *     número leído por teclado.
 *
 * @author Luis José Sánchez
 */
public class S05Ejercicio27 {

  public static void main(String[] args) {

    System.out.print("Introduzca un número entero mayor que 1: ");
    int numeroIntroducido = Integer.parseInt(System.console().readLine());

    int cuenta = 0;
    int suma = 0;

    for (int i = 1; i < numeroIntroducido; i++) {
      if ((i % 3) == 0) {
        System.out.print(i + " ");
        cuenta++;
        suma += i;
      }
    }

    System.out.print("\nDesde 1 hasta " + numeroIntroducido + " hay " + cuenta);
    System.out.println(" múltiplos de 3 y suman " + suma);
  }
}
```

Ejercicio 28

Fichero: S05Ejercicio28.java

```java
/**
 * Aprende Java con Ejercicios (https://leanpub.com/aprendejava)
 *
 * Capítulo 5. Bucles.
 *
 * Ejercicio 28
 *
 * @author Luis José Sánchez
 */
public class S05Ejercicio28 {

  public static void main(String[] args) {

    int numeroIntroducido;

    // Lee un número mayor o igual que 0
    do {
      System.out.print("Por favor, introduzca un número entero: ");
      numeroIntroducido = Integer.parseInt(System.console().readLine());

      if (numeroIntroducido < 0) {
        System.out.println("El número introducido no es correcto.");
      }
    } while (numeroIntroducido < 0);

    int factorial = numeroIntroducido;

    if (numeroIntroducido == 0) {
      System.out.println("El factorial del " + numeroIntroducido + " es 1.");
    } else {
      for (int i = 1; i < numeroIntroducido; i++) {
        factorial *= i;
      }

      System.out.println(numeroIntroducido + "! = " + factorial);
    }
  }
}
```

Ejercicio 29

Fichero: S05Ejercicio29.java

```
/**
 * 5. Bucles
 *
 * 29. Escribe un programa que muestre por pantalla  todos  los números enteros positivos meno\
res
 *     a uno leído por teclado que no sean divisibles entre otro también leído de igual forma.
 *
 * @author Luis José Sánchez
 */
public class S05Ejercicio29 {

  public static void main(String[] args) {

    System.out.print("Introduzca un número entero positivo (relativamente grande): ");
    int numeroGrande = Integer.parseInt(System.console().readLine());

    System.out.print("Introduzca otro número (relativamente pequeño): ");
    int numeroPequeno = Integer.parseInt(System.console().readLine());

    System.out.print("Los números enteros positivos menores que " + numeroGrande );
    System.out.println(" que no son divisibles entre " + numeroPequeno + " son los siguientes:\
");

    int cuenta = 0;
    int suma = 0;

    for (int i = 1; i < numeroGrande; i++) {
      if ((i % numeroPequeno) != 0) {
        System.out.print(i + " ");
      }
    }
  }
}
```

Ejercicio 30

Fichero: S05Ejercicio30.java

```
/**
 * 5. Bucles
 *
 * 30. Realiza una programa que calcule las horas transcurridas entre
 *     dos horas de dos días de la semana. No se tendrán en cuenta los
 *     minutos ni los segundos. El día de la semana se puede pedir como
 *     un número (del 1 al 7) o como una cadena (de “lunes” a “domingo”).
 *     Se debe comprobar que el usuario introduce los datos correctamente
 *     y que el segundo día es posterior al primero.
 *     Ejemplo:
 *     Por favor, introduzca la primera hora.
 *     Día: lunes
 *     Hora: 18
 *     Por favor, introduzca la segunda hora.
 *     Día: martes
 *     Hora: 20
 *     Entre las 18:00h del lunes y las 20:00h del martes hay 26 hora/s.
 *
 * @author Luis José Sánchez
 */
public class S05Ejercicio30 {

  public static void main(String[] args) {

    int primerDia = 0;
    int segundoDia = 0;
    int primeraHora;
    int segundaHora;
    String primerDiaString;
    String segundoDiaString;
    String nombrePrimerDia = "";
    String nombreSegundoDia = "";
    boolean datosCorrectos = true;

    // Recogida de datos /////////////////////////////////////
    do {
      System.out.println("\nPor favor, introduzca la primera hora.");
      System.out.print("Día: ");
      primerDiaString = System.console().readLine();
      System.out.print("Hora: ");
      primeraHora = Integer.parseInt(System.console().readLine());


      switch(primerDiaString) {
        case "lunes":
        case "1":
```

```
      primerDia = 1;
      nombrePrimerDia = "lunes";
      break;
    case "martes":
    case "2":
      primerDia = 2;
      nombrePrimerDia = "martes";
      break;
    case "miércoles":
    case "3":
      primerDia = 3;
      nombrePrimerDia = "miercoles";
      break;
    case "jueves":
    case "4":
      primerDia = 4;
      nombrePrimerDia = "jueves";
      break;
    case "viernes":
    case "5":
      primerDia = 5;
      nombrePrimerDia = "viernes";
      break;
    case "sábado":
    case "6":
      primerDia = 6;
      nombrePrimerDia = "sábado";
      break;
    case "domingo":
    case "7":
      primerDia = 7;
      nombrePrimerDia = "domingo";
      break;
    default:
      primerDia = 0;
  }

  System.out.println("Por favor, introduzca la segunda hora.");
  System.out.print("Día: ");
  segundoDiaString = System.console().readLine();
  System.out.print("Hora: ");
  segundaHora = Integer.parseInt(System.console().readLine());

  switch(segundoDiaString) {
    case "lunes":
    case "1":
```

```
      segundoDia = 1;
      nombreSegundoDia = "lunes";
      break;
    case "martes":
    case "2":
      segundoDia = 2;
      nombreSegundoDia = "martes";
      break;
    case "miércoles":
    case "3":
      segundoDia = 3;
      nombreSegundoDia = "miércoles";
      break;
    case "jueves":
    case "4":
      segundoDia = 4;
      nombreSegundoDia = "jueves";
      break;
    case "viernes":
    case "5":
      segundoDia = 5;
      nombreSegundoDia = "viernes";
      break;
    case "sábado":
    case "6":
      segundoDia = 6;
      nombreSegundoDia = "sábado";
      break;
    case "domingo":
    case "7":
      segundoDia = 7;
      nombreSegundoDia = "domingo";
      break;
    default:
      segundoDia = 0;
  }

  datosCorrectos = true;

  if (segundoDia <= primerDia) {
    System.out.println("El segundo día debe ser posterior al primero.");
    datosCorrectos = false;
  }

  if ((primerDia == 0) || (segundoDia == 0)) {
    System.out.println("No se ha introducido correctamente el día de la semana.");
```

```
        System.out.println("Los días válidos son: lunes, martes, miércoles, jueves y viernes."\
);
        datosCorrectos = false;
      }

      if ((primeraHora < 0) || (primeraHora > 23) || (segundaHora < 0) || (segundaHora > 23)) {
        System.out.println("No se ha introducido correctamente la hora del día.");
        System.out.println("Las horas válidas están entre 0 y 23.");
        datosCorrectos = false;
      }

    } while (!datosCorrectos);
    // Fin de la recogida de datos //////////////////////////

    System.out.print("Entre las " + primeraHora + ":00h del " + nombrePrimerDia);
    System.out.print(" y las " + segundaHora + ":00h del " + nombreSegundoDia);
    System.out.println(" hay " + (((segundoDia * 24) + segundaHora) - ((primerDia * 24) + prim\
eraHora)) + " horas.");

  }
}
```

Ejercicio 31

Fichero: S05Ejercicio31.java

```
/**
 * Aprende Java con Ejercicios (https://leanpub.com/aprendejava
 *
 * Capítulo 5. Bucles.
 *
 * Ejercicio 31
 *
 * @author Luis José Sánchez
 */
public class S05Ejercicio31 {
  public static void main(String[] args) {

    System.out.print("Introduzca la altura de la L: ");
    int altura = Integer.parseInt(System.console().readLine());

    for (int i = 1; i < altura; i++) {
      System.out.println("*");
    }
```

```
    for (int i = 0; i < altura / 2 + 1; i++) {
      System.out.print("* ");
    }

  }
}
```

Ejercicio 32

Fichero: S05Ejercicio32.java

```
/**
 * Aprende Java con Ejercicios (https://leanpub.com/aprendejava)
 *
 * Capítulo 5. Bucles.
 *
 * Ejercicio 32
 *
 * @author Luis José Sánchez
 */
public class S05Ejercicio32 {

  public static void main(String[] args) {

    System.out.print("Por favor, introduzca un número entero positivo: ");
    long numeroIntroducido = Long.parseLong(System.console().readLine());

    // Le da la vueta al número y calcula la longitud
    long numero = numeroIntroducido;
    long volteado = 0;
    int longitud = 0;

    if (numero == 0) {
      longitud = 1;
    }

    while (numero > 0) {
      volteado = (volteado * 10) + (numero % 10);
      numero /= 10;
      longitud++;
    } // while

    // Muestra los dígitos pares
    System.out.print("Dígitos pares: ");
```

```
    int digito;
    int sumaPares = 0;

    for (int i = 0; i < longitud; i++) {

      digito = (int)(volteado % 10);

      if ((digito % 2) == 0) {
        System.out.print(digito + " ");
        sumaPares += digito;
      }

      volteado /= 10;
    }

    // Muestra la suma de los dígitos pares
    System.out.println("\nSuma de los dígitos pares: " + sumaPares);
  }
}
```

Ejercicio 33

Fichero: S05Ejercicio33.java

```
/**
 * Aprende Java con Ejercicios (https://leanpub.com/aprendejava)
 *
 * Capítulo 5. Bucles.
 *
 * Ejercicio 33
 *
 * @author Luis José Sánchez
 */
public class S05Ejercicio33 {
  public static void main(String[] args) {

    System.out.print("Introduzca la altura de la U: ");
    int altura = Integer.parseInt(System.console().readLine());

    // Palos verticales de la U
    for (int i = 1; i < altura; i++) {
      System.out.print("* ");
      for (int j = 2; j < altura; j++) {
        System.out.print("  ");
      }
```

```
      System.out.println("*");
    }

    // Base de la U
    System.out.print("  ");
    for (int i = 2; i < altura; i++) {
      System.out.print("* ");
    }

  }
}
```

Ejercicio 34

Fichero: S05Ejercicio34.java

```
/**
 * Aprende Java con Ejercicios (https://leanpub.com/aprendejava)
 *
 * Capítulo 5. Bucles.
 *
 * Ejercicio 34
 *
 * @author Luis José Sánchez
 */
public class S05Ejercicio34 {

  public static void main(String[] args) {

    // Lectura de dos números por teclado //////////////////////////////////////
    System.out.print("Por favor, introduzca un número: ");
    long numeroIntroducido1 = Long.parseLong(System.console().readLine());

    System.out.print("Introduzca otro número: ");
    long numeroIntroducido2 = Long.parseLong(System.console().readLine());

    // Voltea el primer número y calcula la longitud ///////////////////////////
    // Se da por hecho que los dos números introducidos tienen la misma longitud.
    long numero = numeroIntroducido1;
    long volteado1 = 0;
    int longitud = 0;

    if (numero == 0) {
      longitud = 1;
    }
```

```
while (numero > 0) {
  volteado1 = (volteado1 * 10) + (numero % 10);
  numero /= 10;
  longitud++;
} // while

// Voltea el segundo número /////////////////////////////////////////////////
numero = numeroIntroducido2;
long volteado2 = 0;

while (numero > 0) {
  volteado2 = (volteado2 * 10) + (numero % 10);
  numero /= 10;
} // while

// Recorre los dos números volteados para formar los dos resultados ////////
long resultadoPares = 0;
long resultadoImpares = 0;
int digito;

for (int i = 0; i < longitud; i++) {

  digito = (int)(volteado1 % 10);

  if ((digito % 2) == 0) {
    resultadoPares = resultadoPares * 10 + digito;
  } else {
    resultadoImpares = resultadoImpares * 10 + digito;
  }

  digito = (int)(volteado2 % 10);

  if ((digito % 2) == 0) {
    resultadoPares = resultadoPares * 10 + digito;
  } else {
    resultadoImpares = resultadoImpares * 10 + digito;
  }

  volteado1 /= 10;
  volteado2 /= 10;
}

// Muestra el resultado /////////////////////////////////////////////////////
System.out.println("El número formado por los dígitos pares es: " + resultadoPares);
System.out.println("El número formado por los dígitos impares es: " + resultadoImpares);
```

```
  }
}
```

Ejercicio 35

Fichero: S05Ejercicio35.java

```
/**
 * Aprende Java con Ejercicios (https://leanpub.com/aprendejava)
 *
 * Capítulo 5. Bucles.
 *
 * Ejercicio 35
 *
 * @author Luis José Sánchez
 */
public class S05Ejercicio35 {

  public static void main(String[] args) {

    System.out.print("Por favor, introduzca la altura de la X: ");
    int alturaIntroducida = Integer.parseInt(System.console().readLine());

    int altura = 1;
    int i = 0;
    int espaciosInternos = alturaIntroducida - 1;
    int espaciosPorDelante = 0;

    if ((alturaIntroducida < 3) || (alturaIntroducida % 2 == 0)) {
      System.out.print("Datos incorrectos. Debe introducir una altura impar mayor o igual a 3"\
);
    } else {
      // parte de arriba /////////////////////////////////////
      while (altura < alturaIntroducida / 2 + 1) {

        // inserta espacios delante
        for (i = 1; i <= espaciosPorDelante; i++) {
          System.out.print(" ");
        }

        // pinta la línea
        System.out.print("*");
        for (i = 1; i < espaciosInternos; i++) {
          System.out.print(" ");
        }
```

```
        System.out.print("*");

        System.out.println();
        altura++;
        espaciosPorDelante++;
        espaciosInternos -= 2;
      } // while parte de arriba ///////////////////////

      // parte de abajo ////////////////////////////////////
      espaciosInternos = 0;
      espaciosPorDelante = alturaIntroducida / 2;
      altura = 1;
      while (altura <= alturaIntroducida / 2 + 1) {

        // inserta espacios delante
        for (i = 1; i <= espaciosPorDelante; i++) {
          System.out.print(" ");
        }

        // pinta la línea
        System.out.print("*");
        for (i = 1; i < espaciosInternos; i++) {
          System.out.print(" ");
        }

        if(altura>1) {
          System.out.print("*");
        }

        System.out.println();
        altura++;
        espaciosPorDelante--;
        espaciosInternos+=2;
      } // while parte de abajo //////////////////////
    } // else
  }
}
```

Ejercicio 36

Fichero: S05Ejercicio36.java

```
/**
 * Aprende Java con Ejercicios (https://leanpub.com/aprendejava)
 *
 * Capítulo 5. Bucles.
 *
 * Ejercicio 36
 *
 * @author Luis José Sánchez
 */
public class S05Ejercicio36 {

  public static void main(String[] args) {

    System.out.print("Por favor, introduzca un número entero positivo: ");
    long numeroIntroducido = Long.parseLong(System.console().readLine());

    // Voltea el número introducido.
    long numero = numeroIntroducido;
    long volteado = 0;

    while (numero > 0) {
      volteado = (volteado * 10) + (numero % 10);
      numero /= 10;
    } // while

    if (volteado == numeroIntroducido) {
      System.out.println("El " + numeroIntroducido + " es capicúa");
    } else {
      System.out.println("El " + numeroIntroducido + " no es capicúa");
    }
  }
}
```

Ejercicio 37

Fichero: S05Ejercicio37.java

```java
/**
 * Aprende Java con Ejercicios (https://leanpub.com/aprendejava)
 *
 * Capítulo 5. Bucles.
 *
 * Ejercicio 37
 *
 * @author Luis José Sánchez
 */
public class S05Ejercicio37 {
  public static void main (String[] args) {
    long num;
    do {
      System.out.print("Introduce un número entero positivo: ");
      num = Integer.parseInt(System.console().readLine());
    } while (num < 1);
    System.out.print(num + " = ");

    // Cuenta los números y calcula el reves
    int aux = num;
    int numDig = 0;
    int numReves = 0;
    while (aux > 0){
      numReves = ((numReves*10) + (aux % 10));
      aux /=10;
      numDig++;
    }
    int cifra = 0;
    // Separa las cifras y escribe los palitos
    for (int i = 0; i < numDig; i++){
      cifra = numReves%10;
      numReves /= 10;
      for (int j = 0; j < cifra; j++){
        System.out.print("|");
      }
      if (i < numDig-1){
        System.out.print("-");
      }
    }
  }
}
```

Ejercicio 38

Fichero: S05Ejercicio38.java

```
/**
 * Aprende Java con Ejercicios (https://leanpub.com/aprendejava)
 *
 * Capítulo 5. Bucles.
 *
 * Ejercicio 38
 *
 * @author Luis José Sánchez
 */
public class S05Ejercicio38 {

  public static void main(String[] args) {

    System.out.print("Por favor, introduzca la altura del reloj de arena: ");
    int alturaIntroducida = Integer.parseInt(System.console().readLine());

    int altura = 1;
    int asteriscos = alturaIntroducida;
    int espaciosPorDelante = 0;

    if ((alturaIntroducida < 3) || (alturaIntroducida % 2 == 0)) {
      System.out.print("Datos incorrectos. Debe introducir una altura impar mayor o igual a 3"\
);
    } else {
      // parte de arriba /////////////////////////////////////
      while (altura < alturaIntroducida / 2 + 1) {

        // inserta espacios delante
        for (int i = 1; i <= espaciosPorDelante; i++) {
          System.out.print(" ");
        }

        // pinta la línea
        for (int i = 0; i < asteriscos; i++) {
          System.out.print("*");
        }

        System.out.println();
        altura++;
        espaciosPorDelante++;
        asteriscos -= 2;
      } // while parte de arriba //////////////////////

      // parte de abajo /////////////////////////////////////
      espaciosPorDelante = alturaIntroducida / 2;
      altura = 1;
```

```
      while (altura <= alturaIntroducida / 2 + 1) {

        // inserta espacios delante
        for (int i = 1; i <= espaciosPorDelante; i++) {
          System.out.print(" ");
        }

        // pinta la línea
        for (int i = 0; i < asteriscos; i++) {
          System.out.print("*");
        }

        System.out.println();
        altura++;
        espaciosPorDelante--;
        asteriscos += 2;
      } // while parte de abajo ///////////////////////
    } // else
  }
}
```

Ejercicio 39

Fichero: S05Ejercicio39.java

```
/**
 * Aprende Java con Ejercicios (https://leanpub.com/aprendejava)
 *
 * Capítulo 5. Bucles.
 *
 * Ejercicio 39
 *
 * @author Luis José Sánchez
 */
public class S05Ejercicio39 {
  public static void main(String[] args) {

    int numeroIntroducido;

    System.out.print("Por favor, introduzca un número entero positivo: ");
    numeroIntroducido = Integer.parseInt(System.console().readLine());

    for (int n = 1; n <= numeroIntroducido; n++) {

      int factorial = n;
```

```
      for (int i = 1; i < n; i++) {
        factorial *= i;
      }

      System.out.println(n + "! = " + factorial);
    }
  }
}
```

Ejercicio 40

Fichero: S05Ejercicio40.java

```
/**
 * Aprende Java con Ejercicios (https://leanpub.com/aprendejava)
 *
 * Capítulo 5. Bucles.
 *
 * Ejercicio 40
 *
 * @author Luis José Sánchez
 */
public class S05Ejercicio40 {
  public static void main(String[] args) {

    System.out.print("Por favor, introduzca la altura del rombo: ");
    int alturaIntroducida = Integer.parseInt(System.console().readLine());

    int espaciosInternos = 0;
    int espaciosPorDelante = alturaIntroducida / 2;

    if ((alturaIntroducida < 3) || (alturaIntroducida % 2 == 0)) {
      System.out.print("Datos incorrectos. Debe introducir una altura impar mayor o igual a 3"\
);
    } else {

      int altura = 1;

      // parte de arriba /////////////////////////////////////
      while (altura <= alturaIntroducida / 2 + 1) {

        // inserta espacios delante
        for (int i = 1; i <= espaciosPorDelante; i++) {
          System.out.print(" ");
```

```
      }

      // pinta la línea
      System.out.print("*");
      for (int i = 1; i < espaciosInternos; i++) {
        System.out.print(" ");
      }

      if (altura>1) {
        System.out.print("*");
      }

      System.out.println();
      altura++;
      espaciosPorDelante--;
      espaciosInternos+=2;
    } // while parte de arriba //////////////////////

    // parte de abajo /////////////////////////////////////
    espaciosInternos = alturaIntroducida - 3;
    espaciosPorDelante = 1;
    altura = 0;

    while (altura < alturaIntroducida / 2) {

      // inserta espacios delante
      for (int i = 1; i <= espaciosPorDelante; i++) {
        System.out.print(" ");
      }

      // pinta la línea
      System.out.print("*");
      for (int i = 1; i < espaciosInternos; i++) {
        System.out.print(" ");
      }

      if(altura < alturaIntroducida / 2 - 1) {
        System.out.print("*");
      }

      System.out.println();
      altura++;
      espaciosPorDelante++;
      espaciosInternos -= 2;
    } // while parte de abajo //////////////////////
  } // else
```

```
  }
}
```

Ejercicio 41

Fichero: S05Ejercicio41.java

```
/**
 * Aprende Java con Ejercicios (https://leanpub.com/aprendejava)
 *
 * Capítulo 5. Bucles.
 *
 * Ejercicio 41
 *
 * @author Luis José Sánchez
 */
public class S05Ejercicio41 {
  public static void main(String[] args) {

    System.out.print("Por favor, introduzca un número entero positivo: ");
    long numeroIntroducido = Long.parseLong(System.console().readLine());

    long n = numeroIntroducido;
    int cuentaPares = 0;
    int cuentaImpares = 0;

    while (n > 0) {
      if ((n % 10) % 2 == 0) {
        cuentaPares++;
      } else {
        cuentaImpares++;
      }
      n /= 10;
    }
    System.out.print("El " + numeroIntroducido + " contiene " + cuentaPares + " dígitos pares \
y " + cuentaImpares + " dígitos impares.");
  }
}
```

Ejercicio 42

Fichero: S05Ejercicio42.java

```
/**
 * Aprende Java con Ejercicios (https://leanpub.com/aprendejava)
 *
 * Capítulo 5. Bucles.
 *
 * Ejercicio 42
 *
 * @author Luis José Sánchez
 */
public class S05Ejercicio42 {

  public static void main(String[] args) {

    System.out.print("Por favor, introduzca un número entero positivo: ");
    int numeroIntroducido = Integer.parseInt(System.console().readLine());

    boolean esPrimo;

    for (int n = numeroIntroducido; n < numeroIntroducido + 5; n++) {

      // comprueba si n es primo
      esPrimo = true;
      for (int i = 2; i < n; i++) {
        if (n % i == 0) {
          esPrimo = false;
        }
      }

      // muestra por pantalla si n es primo o no
      if (esPrimo) {
        System.out.println(n + " es primo");
      } else {
        System.out.println(n + " no es primo");
      }
    }
    System.out.println();
  }
}
```

Ejercicio 43

Fichero: S05Ejercicio43.java

```java
/**
 * Aprende Java con Ejercicios (https://leanpub.com/aprendejava)
 *
 * Capítulo 5. Bucles.
 *
 * Ejercicio 43
 *
 * @author Luis José Sánchez
 */
public class S05Ejercicio43 {
  public static void main(String[] args) {

    System.out.print("Por favor, introduzca un número entero positivo: ");
    long numeroIntroducido = Long.parseLong(System.console().readLine());

    System.out.print("Introduzca la posición a partir de la cual quiere partir el número: ");
    long posicion = Long.parseLong(System.console().readLine());

    long numero = numeroIntroducido;

    // calcula la longitud del número
    int longitud = 0;

    do {
      numero /= 10;
      longitud++;
    } while (numero > 0);

    // parte izquierda
    long parteIzquierda = numeroIntroducido / (long)(Math.pow(10, longitud - posicion + 1));

    // parte derecha
    long parteDerecha = numeroIntroducido % (long)(Math.pow(10, longitud - posicion + 1));

    System.out.print("Los números partidos son el " + parteIzquierda + " y  el " + parteDerech\
a + ".");
  }
}
```

Ejercicio 44

Fichero: S05Ejercicio44.java

```java
/**
 * Aprende Java con Ejercicios (https://leanpub.com/aprendejava)
 *
 * Capítulo 5. Bucles.
 *
 * Ejercicio 44
 *
 * @author Luis José Sánchez
 */
public class S05Ejercicio44 {
  public static void main(String[] args) {

    System.out.print("Por favor, introduzca un número entero positivo: ");
    long numeroIntroducido = Long.parseLong(System.console().readLine());

    System.out.print("Introduzca la posición donde quiere insertar: ");
    long posicion = Long.parseLong(System.console().readLine());

    System.out.print("Introduzca el dígito que quiere insertar: ");
    long digito = Long.parseLong(System.console().readLine());

    long numero = numeroIntroducido;

    // calcula la longitud del número
    int longitud = 0;

    do {
      numero /= 10;
      longitud++;
    } while (numero > 0);

    // parte izquierda con el dígito pegado
    long parteIzquierda = numeroIntroducido / (long)(Math.pow(10, longitud - posicion + 1));
    parteIzquierda = parteIzquierda * 10 + digito;

    // parte derecha
    long parteDerecha = numeroIntroducido % (long)(Math.pow(10, longitud - posicion + 1));

    // resultado
    numero = parteIzquierda * (long)(Math.pow(10, longitud - posicion + 1)) + parteDerecha;
    System.out.print("El número resultante es " + numero);
  }
}
```

Ejercicio 45

Fichero: S05Ejercicio45.java

```
/**
 * Aprende Java con Ejercicios (https://leanpub.com/aprendejava)
 *
 * Capítulo 5. Bucles.
 *
 * Ejercicio 45
 *
 * @author Luis José Sánchez
 */
public class S05Ejercicio45 {
  public static void main(String[] args) {

    System.out.print("Por favor, introduzca un número entero positivo: ");
    long numeroIntroducido = Long.parseLong(System.console().readLine());

    System.out.print("Introduzca la posición dentro del número: ");
    long posicion = Long.parseLong(System.console().readLine());

    System.out.print("Introduzca el nuevo dígito: ");
    long digito = Long.parseLong(System.console().readLine());

    long numero = numeroIntroducido;

    // calcula la longitud del número
    int longitud = 0;

    do {
      numero /= 10;
      longitud++;
    } while (numero > 0);

    // parte izquierda con el dígito nuevo
    long parteIzquierda = numeroIntroducido / (long)(Math.pow(10, longitud - posicion + 1));
    parteIzquierda = parteIzquierda * 10 + digito;

    // parte derecha
    long parteDerecha = numeroIntroducido % (long)(Math.pow(10, longitud - posicion));

    // resultado
    numero = parteIzquierda * (long)(Math.pow(10, longitud - posicion)) + parteDerecha;
    System.out.print("El número resultante es " + numero);
  }
```

```
}
```

Ejercicio 46

Fichero: S05Ejercicio46.java

```
/**
 * Aprende Java con Ejercicios (https://leanpub.com/aprendejava)
 *
 * Capítulo 5. Bucles.
 *
 * Ejercicio 46
 *
 * @author Luis José Sánchez
 */
public class S05Ejercicio46 {
  public static void main(String[] args) {

    System.out.print("Por favor, introduzca la anchura del rectángulo (como mínimo 2): ");
    int anchura = Integer.parseInt(System.console().readLine());

    System.out.print("Ahora introduzca la altura (como mínimo 2): ");
    int altura = Integer.parseInt(System.console().readLine());

    if ((anchura < 2) || (altura < 2)) {

      System.out.print("Lo siento, los datos introducidos no son correctos, ");
      System.out.println(" el valor mínimo para la anchura y la altura es 2.");

    } else {

      // Línea superior ////////////////////////////////////////////////////////////
      for (int i = 1; i <= anchura; i++) {
        System.out.print("*");
      }

      // Parte intermedia //////////////////////////////////////////////////////////
      for (int i = 2; i < altura; i ++) {
        System.out.print("\n*");
        for (int espacios = 2; espacios < anchura; espacios++) {
          System.out.print(" ");
        }
        System.out.print("*");
      }
      System.out.println();
```

```
      // Línea inferior /////////////////////////////////////////////////////////
      for (int i = 1; i <= anchura; i++) {
        System.out.print("*");
      }
    } // else
  }
}
```

Ejercicio 47

Fichero: S05Ejercicio47.java

```
import java.util.Scanner;

/**
 * Aprende Java con Ejercicios (https://leanpub.com/aprendejava)
 *
 * Capítulo 5. Bucles.
 *
 * Ejercicio 47
 *
 * @author Luis José Sánchez
 */
public class S05Ejercicio47 {
  public static void main(String[] args) {

    Scanner s = new Scanner(System.in);
    System.out.print("Por favor, introduzca la altura (número impar mayor o igual a 5): ");
    int altura = Integer.parseInt(s.nextLine());

    int alturaAux = (altura - 3) / 2;

    if ((altura % 2 != 0) && (altura >= 5)) {
      System.out.println("MMMMMM");

      for (int i = 0; i < alturaAux; i++) {
        System.out.println("M    M");
      }

      System.out.println("MMMMMM");

      for (int i = 0; i < alturaAux; i++) {
        System.out.println("M    M");
      }
```

```
      System.out.println("MMMMMM");
    } else {
      System.out.println("La altura introducida no es correcta.");
    }
  }
}
```

Ejercicio 48

Fichero: S05Ejercicio48.java

```
import java.util.Scanner;

/**
 * Aprende Java con Ejercicios (https://leanpub.com/aprendejava)
 *
 * Capítulo 5. Bucles.
 *
 * Ejercicio 48
 *
 * @author Luis José Sánchez
 */
public class S05Ejercicio48 {
  public static void main(String[] args) {

    Scanner s = new Scanner(System.in);
    System.out.print("Introduzca un número entero: ");
    long numero = Long.parseLong(s.nextLine());

    long aux;
    boolean encontrado;

    System.out.print("Dígitos que aparecen en el número: ");

    for (int i = 0; i < 10; i++) {
      // Comprueba si i está en el número
      encontrado = false;
      aux = numero;
      while (aux > 0) {
        if (aux % 10 == i) {
          encontrado = true;
        }
        aux /= 10;
      }
```

```
      if (encontrado) {
        System.out.print(i + " ");
      }
    }

    System.out.print("\nDígitos que no aparecen: ");

    for (int i = 0; i < 10; i++) {
      // Comprueba si i está en el número
      encontrado = false;
      aux = numero;
      while (aux > 0) {
        if (aux % 10 == i) {
          encontrado = true;
        }
        aux /= 10;
      }
      if (!encontrado) {
        System.out.print(i + " ");
      }
    }
  }
}
```

Ejercicio 49

Fichero: S05Ejercicio49.java

```
import java.util.Scanner;

/**
 * Aprende Java con Ejercicios (https://leanpub.com/aprendejava)
 *
 * Capítulo 5. Bucles.
 *
 * Ejercicio 49
 *
 * @author Luis José Sánchez
 */
public class S05Ejercicio49 {

  public static void main(String[] args) {

    System.out.println("Por favor, vaya introduciendo números enteros positivos. ");
    System.out.println("Para terminar, introduzca un número primo:");
```

```
    Scanner s = new Scanner(System.in);

    int numero;
    int suma = 0;
    int cuentaNumeros = 0;
    int maximo = Integer.MIN_VALUE;
    int minimo = Integer.MAX_VALUE;
    boolean esPrimo;

    do {
      numero = Integer.parseInt(s.nextLine());

      // comprueba si el número introducido es primo
      esPrimo = true;
      for (int i = 2; i < numero; i++) {
        if ((numero % i) == 0) {
          esPrimo = false;
        }
      }

      // si no es primo, se contabiliza
      if (!esPrimo) {
        suma += numero;
        cuentaNumeros++;

        maximo = numero > maximo ? numero : maximo;
        minimo = numero < minimo ? numero : minimo;
      }

    } while (!esPrimo);

    System.out.println("Ha introducido " + cuentaNumeros + " números no primos.");
    System.out.println("Máximo: " + maximo);
    System.out.println("Mínimo: " + minimo);
    System.out.println("Media: " + (double)suma / cuentaNumeros);
  }
}
```

Ejercicio 50

Fichero: S05Ejercicio50.java

```java
import java.util.Scanner;

/**
 * Aprende Java con Ejercicios
 * https://leanpub.com/aprendejava
 *
 * Capítulo 5. Bucles.
 *
 * Ejercicio 50
 *
 * @author Luis José Sánchez
 */
public class S05Ejercicio50 {

  public static void main(String[] args) {

    Scanner s = new Scanner(System.in);
    System.out.print("Introduzca la altura (5 como mínimo): ");
    int altura = Integer.parseInt(s.nextLine());
    System.out.print("Introduzca el número de espacios entre los números (1 como mínimo): ");
    int espacios = Integer.parseInt(s.nextLine());

    // Parte superior del 5 //////////////////////////
    // Fila 1
    System.out.print("*");
    for (int i = 0; i < espacios; i++) {
      System.out.print(" ");
    }
    System.out.print("****");
    for (int i = 0; i < espacios; i++) {
      System.out.print(" ");
    }
    System.out.println("****");

    // Fila 2
    System.out.print("*");
    for (int i = 0; i < espacios; i++) {
      System.out.print(" ");
    }
    System.out.print("*   ");
    for (int i = 0; i < espacios; i++) {
      System.out.print(" ");
    }
    System.out.println("*");

    // Fila 3 (igual que la fila 1)
```

```
    System.out.print("*");
    for (int i = 0; i < espacios; i++) {
      System.out.print(" ");
    }
    System.out.print("****");
    for (int i = 0; i < espacios; i++) {
      System.out.print(" ");
    }
    System.out.println("****");

    // Parte inferior del 5 //////////////////////////
    // Filas variables
    for (int fila = 0; fila < altura - 4; fila++) {
      System.out.print("*");
      for (int i = 0; i < espacios; i++) {
        System.out.print(" ");
      }
      System.out.print("   *");
      for (int i = 0; i < espacios; i++) {
        System.out.print(" ");
      }
      System.out.println("   *");
    }

    // Última fila (igual que la primera)
    System.out.print("*");
    for (int i = 0; i < espacios; i++) {
      System.out.print(" ");
    }
    System.out.print("****");
    for (int i = 0; i < espacios; i++) {
      System.out.print(" ");
    }
    System.out.println("****");
  }
}
```

Ejercicio 51

Fichero: S05Ejercicio51.java

```java
import java.util.Scanner;

/**
 * Aprende Java con Ejercicios
 * https://leanpub.com/aprendejava
 *
 * Capítulo 5. Bucles.
 *
 * Ejercicio 51
 *
 * @author Luis José Sánchez
 */
public class S05Ejercicio51 {

  public static void main(String[] args) {

    Scanner s = new Scanner(System.in);
    System.out.print("Introduzca un número entero (mayor que cero): ");
    int numeroIntroducido = Integer.parseInt(s.nextLine());

    int numero = numeroIntroducido;
    int volteado = 0;
    boolean numeroComido = false;

    // Al mismo tiempo que se le da la vuelta al número
    // se le quitan los 0s y 8s
    while (numero > 0) {
      if ((numero % 10 != 0) && (numero % 10 != 8)) {
        volteado = (volteado * 10) + (numero % 10);
      } else {
        numeroComido = true;
      }
      numero /= 10;
    }

    // Se le vuelve a dar la vuelta al número
    numero = volteado;
    volteado = 0;

    while (numero > 0) {
      volteado = (volteado * 10) + (numero % 10);
      numero /= 10;
    }

    if (numeroComido) {
      System.out.print("Después de haber sido comido por el gusano numérico");
```

```
      System.out.println(" se queda en " + volteado);
    } else {
      System.out.println("El gusano numérico no se ha comido ningún dígito.");
    }
  }
}
```

Ejercicio 52

Fichero: S05Ejercicio52.java

```
import java.util.Scanner;

public class S05Ejercicio52 {

  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    System.out.print("Introduzca un número: ");
    long numeroIntroducido = Long.parseLong(s.nextLine());

    long copia = numeroIntroducido;
    int longitud = 0;

    while (copia > 0) {
      copia /= 10;
      longitud++;
    }

    int primero =  (int)(numeroIntroducido / Math.pow(10, longitud - 1));
    long aux = (long)(primero * Math.pow(10, longitud - 1));
    long resultado = (numeroIntroducido - aux) * 10 + primero;
    System.out.println("El número resultado es " + resultado);
  }

}
```

Ejercicio 53

Fichero: S05Ejercicio53.java

```
import java.util.Scanner;

public class S05Ejercicio53 {

  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    System.out.print("Introduzca la altura de la figura: ");
    int altura = Integer.parseInt(s.nextLine());

    for (int i = 0; i < altura; i++) {
      for (int j = 0; j < altura - i; j++) {
        System.out.print("*");
      }
      System.out.println();
    }
  }

}
```

Ejercicio 54

Fichero: S05Ejercicio54.java

```
import java.util.Scanner;

public class S05Ejercicio54 {

  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    System.out.print("Introduzca la altura de la figura: ");
    int altura = Integer.parseInt(s.nextLine());

    // Primera línea
    for (int i = 0; i < altura; i++) {
      System.out.print("*");
    }
    System.out.println();

    // Parte intermedia
    for (int i = 1; i < altura - 1; i++) {
      System.out.print("*");
      for (int j = 0; j < altura - i - 2; j++) {
        System.out.print(" ");
      }
      System.out.println("*");
```

```
    }

    // Vértice inferior
    System.out.println("*");
  }

}
```

Ejercicio 55

Fichero: S05Ejercicio55.java

```
import java.util.Scanner;

public class S05Ejercicio55 {

  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    System.out.print("Introduzca un número: ");
    long numeroIntroducido = Long.parseLong(s.nextLine());

    long copia = numeroIntroducido;
    int longitud = 0;

    while (copia > 0) {
      copia /= 10;
      longitud++;
    }

    int ultimo = (int) (numeroIntroducido % 10);
    long aux = numeroIntroducido / 10;
    long resultado = ultimo * (long) Math.pow(10, longitud - 1) + aux;
    System.out.println("El número resultado es " + resultado);
  }

}
```

Ejercicio 56

Fichero: S05Ejercicio56.java

```
import java.util.Scanner;

public class S05Ejercicio56 {

  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    System.out.print("Introduzca la altura de la figura: ");
    int altura = Integer.parseInt(s.nextLine());

    int espacios = 0;

    for (int i = 0; i < altura; i++) {
      for (int j = 0; j < espacios; j++) {
        System.out.print(" ");
      }

      for (int j = 0; j < altura - i; j++) {
        System.out.print("*");
      }

      System.out.println();

      espacios++;
    }
  }

}
```

Ejercicio 57

Fichero: S05Ejercicio57.java

```
import java.util.Scanner;

public class S05Ejercicio57 {

  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    System.out.print("Introduzca la altura de la figura: ");
    int altura = Integer.parseInt(s.nextLine());

    // Primera línea
    for (int i = 0; i < altura; i++) {
      System.out.print("*");
    }
```

```
    System.out.println();

    // Parte intermedia
    int espacios = 1;
    for (int i = 1; i < altura - 1; i++) {
      for (int j = 0; j < espacios; j++) {
        System.out.print(" ");
      }
      System.out.print("*");
      for (int j = 0; j < altura - i - 2; j++) {
        System.out.print(" ");
      }
      System.out.println("*");
      espacios++;
    }

    // Vértice inferior
    for (int i = 0; i < espacios; i++) {
        System.out.print(" ");
    }
    System.out.println("*");
  }

}
```

Ejercicio 58

Fichero: S05Ejercicio58.java

```
import java.util.Scanner;

public class S05Ejercicio58 {

  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    System.out.print("Introduzca un número: ");
    long numeroIntroducido = Long.parseLong(s.nextLine());

    int suma = 0;
    int longitud = 0;

    while (numeroIntroducido > 0) {
      suma += numeroIntroducido % 10;
      numeroIntroducido /= 10;
      longitud++;
```

```
    }

    System.out.println("La media de sus dígitos es " + (double)suma / longitud);
  }

}
```

Ejercicio 59

Fichero: S05Ejercicio59.java

```
import java.util.Scanner;

public class S05Ejercicio59 {

  public static void main(String[] args) {

    Scanner s = new Scanner(System.in);

    System.out.print("Por favor, introduzca la altura del árbol: ");
    int alturaIntroducida = Integer.parseInt(s.nextLine());

    int espaciosPorDelante = alturaIntroducida - 3;
    int espaciosInternos = 0;

    // Pinta la estrella del árbol
    for (int i = 1; i <= espaciosPorDelante; i++) {
      System.out.print(" ");
    }
    System.out.println("*");

    // Empieza por la segunda fila ya que en la primera está la estrella
    int altura = 2;

    while (altura < alturaIntroducida - 1) {

      // inserta espacios delante
      for (int i = 1; i <= espaciosPorDelante; i++) {
        System.out.print(" ");
      }

      // pinta la línea
      System.out.print("#");
      for (int i = 1; i < espaciosInternos; i++) {
        System.out.print(" ");
```

```
      }

      if (altura > 2) {
        System.out.print("#");
      }

      System.out.println();
      altura++;
      espaciosPorDelante--;
      espaciosInternos += 2;
    } // while ///////////////////////////

    // base de la pirámide
    for (int i = 1; i < altura * 2 - 2; i++) {
      System.out.print("#");
    }
    System.out.println();

    // Pinta el tronco
    espaciosPorDelante = alturaIntroducida - 3;

    for (int i = 1; i <= espaciosPorDelante; i++) {
      System.out.print(" ");
    }
    System.out.println("#");
  }

}
```

Ejercicio 60

Fichero: S05Ejercicio60.java

```
public class S05Ejercicio60 {
  public static void main(String[] args) {

    System.out.print("Introduzca la altura de los calcetines: ");
    int altura = Integer.parseInt(System.console().readLine());

    for (int i = 0; i < altura - 2; i++) {
      System.out.println("***     ***");
    }

    for (int i = 0; i < 2; i++) {
      System.out.println("******  ******");
```

```
    }

  }
}
```

Ejercicio 61

Fichero: S05Ejercicio61.java

```
import java.util.Scanner;

public class S05Ejercicio61 {

  public static void main(String[] args) {

    Scanner s = new Scanner(System.in);
    System.out.print("Introduzca la altura de la V (un número mayor o igual a 3): ");
    int alturaIntroducida = Integer.parseInt(s.nextLine());

    int altura = 0;
    int i = 0;
    int espaciosInternos = alturaIntroducida * 2 - 1;
    int espaciosPorDelante = 0;

    if (alturaIntroducida < 3) {
      System.out.println("La altura debe ser mayor o igual a 3.");
    } else {
      while (altura < alturaIntroducida) {
        // inserta espacios delante
        for (i = 1; i <= espaciosPorDelante; i++) {
          System.out.print(" ");
        }

        // pinta la línea
        System.out.print("***");
        for (i = 1; i < espaciosInternos; i++) {
          System.out.print(" ");
        }
        System.out.println("***");

        altura++;
        espaciosPorDelante++;
        espaciosInternos -= 2;
      } // while
    } // else
```

```
  }

}
```

Ejercicio 62

Fichero: S05Ejercicio62.java

```
import java.util.Scanner;

public class S05Ejercicio62 {

  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    System.out.print("Introduzca un número: ");
    long numeroIntroducido = Long.parseLong(s.nextLine());
    long copia = numeroIntroducido;

    int afortunados = 0;
    int noAfortunados = 0;

    while (numeroIntroducido > 0) {
          int digito = (int)(numeroIntroducido % 10);
      if ((digito == 3) || (digito == 7) || (digito == 8) || (digito == 9)) {
                afortunados++;
      } else {
                noAfortunados++;
          }
          numeroIntroducido /= 10;
    }

        if (afortunados > noAfortunados) {
          System.out.println("El " + copia + " es un número afortunado.");
        } else {
          System.out.println("El " + copia + " no es un número afortunado.");
    }

  }

}
```

Ejercicio 63

Fichero: S05Ejercicio63.java

```
import java.util.Scanner;

public class S05Ejercicio63 {

  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);

    System.out.print("Introduzca la altura de la primera pirámide: ");
    int alturaP1 = Integer.parseInt(s.nextLine());

    System.out.print("Introduzca la altura de la segunda pirámide: ");
    int alturaP2 = Integer.parseInt(s.nextLine());

    int alturaMaxima = Math.max(alturaP1, alturaP2);
    int espaciosP1 = alturaP1 - 1;
    int espaciosP2 = alturaP2 - 1;
    int baseP1 = alturaP1 * 2 - 1;
    int baseP2 = alturaP2 * 2 - 1;
    int asteriscosP1 = 1;
    int asteriscosP2 = 1;

    for (int i = alturaMaxima; i > 0; i--) {

      // Pirámide 1

      if (i > alturaP1) {
        // Si la pirámide 1 queda por debajo, pinta el hueco.
        for (int j = 0; j <= baseP1; j++) {
          System.out.print(" ");
        }
      } else {
        // Espacios por delante de la pirámide 1
        for (int j = 0; j < espaciosP1; j++) {
          System.out.print(" ");
        }

        // Pirámide 1
        for (int j = 0; j < asteriscosP1; j++) {
          System.out.print("*");
        }

        // Espacios por detrás de la pirámide 1
        for (int j = 0; j <= espaciosP1; j++) {
          System.out.print(" ");
        }
```

```
        espaciosP1--;
        asteriscosP1 += 2;
      }

      // Pirámide 2

      if (i > alturaP2) {
        // Si la pirámide 2 queda por debajo, pinta el hueco.
        for (int j = 0; j < baseP2; j++) {
          System.out.print(" ");
        }
      } else {
        // Espacios por delante de la pirámide 1
        for (int j = 0; j < espaciosP2; j++) {
          System.out.print(" ");
        }

        // Pirámide 1
        for (int j = 0; j < asteriscosP2; j++) {
          System.out.print("*");
        }

        // Espacios por detrás de la pirámide 1
        for (int j = 0; j <= espaciosP1; j++) {
          System.out.print(" ");
        }

        espaciosP2--;
        asteriscosP2 += 2;
      }
      System.out.println();
    }
  }

}
```

Ejercicio 64

Fichero: S05Ejercicio64.java

```
import java.util.Scanner;

public class S05Ejercicio64 {

  public static void main(String[] args) {

    int ancho = 6;
    int alto = 3;

    Scanner s = new Scanner(System.in);

    int opcion = 0;

    do {
      // Pinta el rectángulo
      for (int i = 0; i < ancho; i++) {
          System.out.print("*");
      }
      System.out.println();

      for (int i = 0; i < alto - 2; i++) {
        System.out.print("*");
        for (int j = 0; j < ancho - 2; j++) {
          System.out.print(" ");
        }
        System.out.println("*");
      }

      for (int i = 0; i < ancho; i++) {
          System.out.print("*");
      }
      System.out.println();

      // Menú
      System.out.println("1. Agrandarlo");
      System.out.println("2. Achicarlo");
      System.out.println("3. Cambiar la orientación");
      System.out.println("4. Salir");
      System.out.print("Indique qué quiere hacer con el rectángulo: ");
      opcion = Integer.parseInt(s.nextLine());

      switch(opcion) {
        case 1:
          ancho++;
          alto++;
          break;
```

```
        case 2:
          if ((ancho > 2) && (alto > 2)) {
            ancho--;
            alto--;
          }
          break;
        case 3:
          int aux = ancho;
          ancho = alto;
          alto = aux;
          break;
        default:
      }
      System.out.println();
    } while (opcion != 4);
  }

}
```

Ejercicio 65

Fichero: S05Ejercicio65.java

```
import java.util.Scanner;

public class S05Ejercicio65 {

  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    System.out.print("Introduzca la altura de la A (un número mayor o igual a 3): ");
    int altura = Integer.parseInt(s.nextLine());

    if (altura < 3) {
      System.out.println("La altura introducida no es correcta.");
    } else {
      System.out.print("Introduzca la fila del palito horizontal (entre 2 y " + (altura - 1) +\
 "): ");
      int palito = Integer.parseInt(s.nextLine());

      if ((palito < 2) || (palito >= altura)) {
        System.out.println("La fila introducida no es correcta.");
      } else {
        // Pinta la letra A
        int espaciosIzq = altura - 1;
        int espaciosCentro = 1;
```

```
        for (int fila = 1; fila <= altura; fila++) {
          repiteCaracter(" ", espaciosIzq--);
          System.out.print("*");
          if (fila > 1) {
            if (fila == palito) {
              repiteCaracter("*", espaciosCentro);
            } else {
              repiteCaracter(" ", espaciosCentro);
            }

            espaciosCentro+=2;
            System.out.print("*");
          }
          System.out.println();
        }

      }
    }
  }

  public static void repiteCaracter(String c, int n) {
    for (int i = 0; i < n; i++) {
      System.out.print(c);
    }
  }

}
```

Ejercicio 66

Fichero: S05Ejercicio66.java

```
import java.util.Scanner;

public class S05Ejercicio66 {

  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);

    System.out.print("Por favor, introduzca la altura de la figura: ");
    int altura = Integer.parseInt(s.nextLine());

    if ((altura < 3) || (altura % 2 == 0)) {
      System.out.println("La altura no es correcta, debe ser un número impar mayor o igual que\
```

```
3.");
    } else {
      System.out.println();
      for (int i = 0; i < altura / 2; i++) {
        for (int j = 0; j < i; j++) {
          System.out.print(" ");
        }
        System.out.println("*    *");
      }

      for (int i = altura / 2; i >= 0; i--) {
        for (int j = 0; j < i; j++) {
          System.out.print(" ");
        }
        System.out.println("*    *");
      }
    }
  }

}
```

Ejercicio 67

Fichero: S05Ejercicio67.java

```
import java.util.Scanner;

public class S05Ejercicio67 {

  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);

    System.out.print("Introduzca el número de escalones: ");
    int escalones = Integer.parseInt(s.nextLine());

    System.out.print("Introduzca la altura de cada escalón: ");
    int alturaEscalon = Integer.parseInt(s.nextLine());

    for (int i = 1; i <= escalones; i++) {
      for (int j = 1; j <= alturaEscalon; j++) {
        for (int k = 0; k < i; k++) {
          System.out.print("****");
        }
        System.out.println();
      }
```

```
    }
  }

}
```

Ejercicio 68

Fichero: S05Ejercicio68.java

```
import java.util.Scanner;

public class S05Ejercicio68 {

  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    System.out.print("Por favor, introduzca un número: ");
    long numeroIntroducido = Long.parseLong(s.nextLine());
    long copia = numeroIntroducido;

    long resultado = 0;
    int cuentaDigitos = 0;

    while (numeroIntroducido > 0) {
      int digito = (int) (numeroIntroducido % 10);
      int nuevoDigito = digito % 2 == 1 ? --digito : ++digito;
      resultado = (long) (nuevoDigito * Math.pow(10, cuentaDigitos) + resultado);
      numeroIntroducido /= 10;
      cuentaDigitos++;
    }

    System.out.println("Dislocando el " + copia + " sale el " + resultado);
  }

}
```

Ejercicio 69

Fichero: S05Ejercicio69.java

```
import java.util.Scanner;

public class S05Ejercicio69 {

  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);

    System.out.print("Introduzca la altura de la pirámide maya: ");
    int alturaIntroducida = Integer.parseInt(s.nextLine());

    int planta = 1;
    int longitudDeLinea = 1;
    int espacios = alturaIntroducida - 1;

    while (planta <= alturaIntroducida) {

      // Izquierda
      for (int i = 1; i <= espacios; i++) {
        System.out.print(' ');
      }
      for (int i = 0; i <= longitudDeLinea / 2; i++) {
        System.out.print('*');
      }

      // Centro
      System.out.print(planta % 2 == 1 ? "****" : "    ");

      // Derecha
      for (int i = 0; i <= longitudDeLinea / 2; i++) {
        System.out.print('*');
      }
      System.out.println();

      planta++;
      espacios--;
      longitudDeLinea += 2;
    }
  }
}
```

Números aleatorios

Ejercicio 1

Fichero: S06Ejercicio01.java

```
/**
 * 6. Números aleatorios
 *
 * 1. Escribe un programa que muestre la tirada de tres dados.
 *    Se debe mostrar también la suma total (los puntos que suman
 *    entre los tres dados).
 *
 * @author Luis José Sánchez
 */
public class S06Ejercicio01 {
  public static void main(String[] args) {

    int tirada;
    int suma = 0;

    System.out.print("Tirada de tres dados: ");

    for(int i = 0; i < 3; i++) {
      tirada = (int)(Math.random() * 6) + 1;
      System.out.print(tirada + " ");
      suma += tirada;
    }

    System.out.println("\nSuma: " + suma);
  }
}
```

Ejercicio 2

Fichero: S06Ejercicio02.java

```
/**
 * 6. Números aleatorios
 *
 * 2. Realiza un programa que muestre al azar el nombre de una carta de
 *    la baraja francesa. Esta baraja está dividida en cuatro palos: picas,
 *    corazones, diamantes y tréboles. Cada palo está formado por 13 cartas,
 *    de las cuales 9 cartas son numerales y 4 literales: 2, 3, 4, 5, 6, 7,
 *    8, 9, 10, J, Q, K y A (que sería el 1). Para convertir un número en
 *    una cadena de caracteres podemos usar String.valueOf(n).
 *
 * @author Luis José Sánchez
 */
public class S06Ejercicio02 {
  public static void main(String[] args) {

    String palo = "";
    String carta = "";

    int numeroPalo = (int)(Math.random()*4) + 1;

    switch(numeroPalo) {
      case 1:
        palo = "picas";
        break;
      case 2:
        palo = "corazones";
        break;
      case 3:
        palo = "diamantes";
        break;
      case 4:
        palo = "tréboles";
      default:
    }

    int numeroCarta = (int)(Math.random()*13) + 1;

    switch(numeroCarta) {
      case 1:
        carta = "As";
        break;
      case 11:
        carta = "J";
        break;
      case 12:
        carta = "Q";
```

```
        break;
      case 13:
        carta = "K";
        break;
      default:
        carta = String.valueOf(numeroCarta);
    }

    System.out.println(carta + " de " + palo);
  }
}
```

Ejercicio 3

Fichero: S06Ejercicio03.java

```
/**
 * 6. Números aleatorios
 *
 * 3. Igual que el ejercicio anterior pero con la baraja española. Se utilizará la baraja de
 *    48 cartas: 2, 3, 4, 5, 6, 7, 8, 9, sota, caballo, rey y as.
 *
 * @author Luis José Sánchez
 */
public class S06Ejercicio03 {

  public static void main(String[] args) {

    String palo = "";
    String carta = "";

    switch((int)(Math.random()*4)) {
      case 0:
        palo = "oros";
        break;
      case 1:
        palo = "copas";
        break;
      case 2:
        palo = "bastos";
        break;
      case 3:
        palo = "espadas";
      default:
    }
```

```
    int numeroCarta = (int)(Math.random()*11) + 1;

    switch(numeroCarta) {
      case 1:
        carta = "As";
        break;
      case 8:
        carta = "Sota";
        break;
      case 9:
        carta = "Caballo";
        break;
      case 10:
        carta = "Rey";
        break;
      default:
        carta = String.valueOf(numeroCarta);
    }

    System.out.println(carta + " de " + palo);
  }
}
```

Ejercicio 4

Fichero: S06Ejercicio04.java

```
/**
 * 6. Números aleatorios
 *
 * 4. Muestra 20 números enteros aleatorios entre 0 y 10 (ambos incluidos)
 *    separados por espacios.
 *
 * @author Luis José Sánchez
 */
public class S06Ejercicio04 {
  public static void main(String[] args) {

    for (int i = 0; i < 20; i++) {
      System.out.print((int)(Math.random()*11) + " ");
    }

    System.out.println();
  }
}
```

Ejercicio 5

Fichero: S06Ejercicio05.java

```
/**
 * 6. Números aleatorios
 *
 * 5. Muestra 50 números enteros aleatorios entre 100 y 199 (ambos incluidos)
 *    separados por espacios. Muestra el máximo, el mínimo y la media de esos números.
 *
 * @author Luis José Sánchez
 */
public class S06Ejercicio05 {
  public static void main(String[] args) {

    int maximo = 100;
    int minimo = 199;
    int suma = 0;
    int n;

    for (int i = 0; i < 50; i++) {
      n = (int)(Math.random()*100) + 100;
      System.out.print(n + " ");
      suma += n;

      if (n < minimo) {
        minimo = n;
      }

      if (n > maximo) {
        maximo = n;
      }
    }

    System.out.println("\nMínimo: " + minimo + "\nMáximo: " + maximo + "\nMedia: " + suma / 50\
);
  }
}
```

Ejercicio 6

Fichero: S06Ejercicio06.java

```
/**
 * 6. Números aleatorios
 *
 * 6. Escribe un programa que piense un número al azar entre 0 y 100. El usuario debe
 *    adivinarlo y tiene para ello 5 oportunidades. Después de cada intento fallido, el
 *    programa dirá cuántas oportunidades quedan y si el número introducido es menor o
 *    mayor que el que ha pensado.
 *
 * @author Luis José Sánchez
 */
public class S06Ejercicio06 {
  public static void main(String[] args) {

    int oportunidades = 5;
    int numeroIntroducido;
    boolean acertado = false;
    int numeroMisterioso = (int)(Math.random() * 101);

    System.out.println("Estoy pensando un número del 0 al 100, intenta adivinarlo. Tienes 5 op\
ortunidades.");

    do {
      System.out.print("Introduce un número: ");
      numeroIntroducido = Integer.parseInt(System.console().readLine());
      oportunidades--;

      if ( (numeroIntroducido > numeroMisterioso) && (oportunidades > 0) ){
        System.out.println("El número que estoy pensando es menor que " + numeroIntroducido);
        System.out.println("Te quedan " + oportunidades + " oportunidades.");
      }

      if ( (numeroIntroducido < numeroMisterioso) && (oportunidades > 0) ){
        System.out.println("El número que estoy pensando es mayor que " + numeroIntroducido);
        System.out.println("Te quedan " + oportunidades + " oportunidades.");
      }

      if (numeroIntroducido == numeroMisterioso) {
        acertado = true;
        System.out.println("¡Enhorabuena! ¡has acertado!");
      }
    } while (!acertado && (oportunidades > 0));

    if (!acertado) {
      System.out.println("Lo siento, no has acertado, el número que estaba pensando era el " +\
 numeroMisterioso);
    }
```

```
  }
}
```

Ejercicio 7

Fichero: S06Ejercicio07.java

```
/**
 * 6. Números aleatorios
 *
 * 7. Escribe un programa que muestre tres apuestas de la quiniela en
 *    tres columnas para los partidos y el pleno al quince (15 filas).
 *
 * @author Luis José Sánchez
 */
public class S06Ejercicio07 {
  public static void main(String[] args) {

    int resultadoPartido;
    int columnas = 3;

    for (int fila = 1; fila <= 15; fila++) {
      System.out.printf("%4d. |", fila);

      if (fila == 15) {
        columnas = 1;
      }

      for (int apuesta = 1; apuesta <= columnas; apuesta++) {
        resultadoPartido = (int)(Math.random() * 3) + 1;
        switch(resultadoPartido) {
          case 1:
            System.out.print("1  |");
            break;
          case 2:
            System.out.print("  2|");
            break;
          case 3:
            System.out.print(" X |");
          default:
        }
      }
      System.out.println();
    }
  }
}
```

Ejercicio 8

Fichero: S06Ejercicio08.java

```
/**
 * 6. Números aleatorios
 *
 * 8. Modifica el programa anterior para que la probabilidad de que salga un "1" sea de 1/2,
 *    la probabilidad de que salga "x" sea de 1/3 y la probabilidad de que salga "2" sea de 1/\
6.
 *    Nótese que 1/2 = 3/6 y 1/3 = 2/6.
 *
 * @author Luis José Sánchez
 */
public class S06Ejercicio08 {
  public static void main(String[] args) {

    int resultadoPartido;
    int columnas = 3;

    for (int fila = 1; fila <= 15; fila++) {
      System.out.printf("%4d. |", fila);

      if (fila == 15) {
        columnas = 1;
      }

      for (int apuesta = 1; apuesta <= columnas; apuesta++) {
        resultadoPartido = (int)(Math.random() * 6) + 1;
        switch(resultadoPartido) {
          case 1:
          case 2:
          case 3:
            System.out.print("1  |");
            break;
          case 4:
          case 5:
            System.out.print(" X |");
            break;
          case 6:
            System.out.print("  2|");
          default:
        }
      }
      System.out.println();
    }
```

```
  }
}
```

Ejercicio 9

Fichero: S06Ejercicio09.java

```
/**
 * 6. Números aleatorios
 *
 * 9. Realiza un programa que vaya generando números aleatorios pares
 *    entre 0 y 100 y que no termine hasta que no saque el número 24. El
 *    programa deberá decir al final cuántos números se han generado.
 *
 * @author Luis José Sánchez
 */
public class S06Ejercicio09 {
  public static void main(String[] args) {

    int n = 0;
    int cuentaNumeros = 0;

    while (n != 24) {
      n = (int)(Math.random() * 51) * 2;
      System.out.print(n + " ");
      cuentaNumeros++;
    }
    System.out.println("\nSe han generado " + cuentaNumeros + " números.");
  }
}
```

Ejercicio 10

Fichero: S06Ejercicio10.java

```java
/**
 * 6. Números aleatorios
 *
 * 10. Realiza un programa que pinte por pantalla diez líneas formadas por caracteres.
 *     El carácter con el que se pinta cada línea se elige de forma aleatoria entre
 *     uno de los siguientes: *, -, =, ., |, @. Las líneas deben tener una longitud
 *     aleatoria entre 1 y 40 caracteres.
 *
 * @author Luis José Sánchez
 */
public class S06Ejercicio10 {

  public static void main(String[] args) {

    int r;
    int longitud;
    String relleno = "";

    for(int i = 1; i <= 10; i++) {

      longitud = (int)(Math.random() * 40) + 1;
      r = (int)(Math.random() * 6);

      switch(r) {
        case 0:
          relleno = "*";
          break;
        case 1:
          relleno = "-";
          break;
        case 2:
          relleno = "=";
          break;
        case 3:
          relleno = ".";
          break;
        case 4:
          relleno = "|";
          break;
        case 5:
          relleno = "@";
          break;
        default:
      }

      // pinta la línea
```

```
      for(int j = 1; j <= longitud; j++) {
        System.out.print(relleno);
      }
      System.out.println();
    }
  }
}
```

Ejercicio 11

Fichero: S06Ejercicio11.java

```
/**
 * 6. Números aleatorios
 *
 * 11. Escribe un programa que muestre 20 notas generadas al azar. Las notas deben aparecer de\
 la
 *     forma: suspenso, suficiente, bien, notable o sobresaliente. Al final aparecerá el númer\
o de
 *     suspensos, el número de suficientes, el número de bienes, etc.
 *
 * @author Luis José Sánchez
 */

public class S06Ejercicio11 {

  public static void main(String[] args) {

    int nota;
    int suspensos = 0;
    int suficientes = 0;
    int bienes = 0;
    int notables = 0;
    int sobresalientes = 0;

    for(int i = 0; i < 20; i++) {

      nota = (int)(Math.random() * 5);

      switch(nota) {
        case 0:
          System.out.print("suspenso ");
          suspensos++;
          break;
        case 1:
```

```
          System.out.print("suficiente ");
          suficientes++;
          break;
        case 2:
          System.out.print("bien ");
          bienes++;
          break;
        case 3:
          System.out.print("notable ");
          notables++;
          break;
        case 4:
          System.out.print("sobresaliente ");
          sobresalientes++;
          break;
        default:
      }

    }

    System.out.println("\nNº de suspensos: " + suspensos);
    System.out.println("Nº de suficientes: " + suficientes);
    System.out.println("Nº de bienes: " + bienes);
    System.out.println("Nº de notables: " + notables);
    System.out.println("Nº de sobresalientes: " + sobresalientes);
  }
}
```

Ejercicio 12

Fichero: S06Ejercicio12.java

```
/**
 * 6. Números aleatorios
 *
 * 12. Realiza un programa que llene la pantalla de caracteres aleatorios (a lo Matrix) con el
 *     código ascii entre el 32 y el 126. Puedes hacer casting con (char) para convertir un
 *     entero en un carácter.
 *
 * @author Luis José Sánchez
 */
public class S06Ejercicio12 {

  public static void main(String[] args)
    throws InterruptedException {
```

```
    int linea = 0;

    System.out.print("\033[32m"); // pinta en verde

    for(int i = 0; i < 8000; i++) {
      System.out.print((char)(Math.random() * (126 - 32 + 1) + 32));

      if (linea++ == 60) {
        linea = 0;
        Thread.sleep(50);
        System.out.println();
      }
    }
  }
}
```

Ejercicio 13

Fichero: S06Ejercicio13.java

```
/**
 * 6. Números aleatorios
 *
 * 13. Escribe un programa que simule la tirada de dos dados. El programa deberá
 *     continuar tirando los dados una y otra vez hasta que en alguna tirada los
 *     dos dados tengan el mismo valor.
 *
 * @author Luis José Sánchez
 */
public class S06Ejercicio13 {

  public static void main(String[] args) {

    int dado1, dado2;

    do {
      dado1 = (int)(Math.random() * 6) + 1;
      dado2 = (int)(Math.random() * 6) + 1;
      System.out.println(dado1 + " " + dado2);
    } while (dado1 != dado2);
  }
}
```

Ejercicio 14

Fichero: S06Ejercicio14.java

```
/**
 * 6. Números aleatorios
 *
 * 14. Escribe un programa que piense un número al azar entre 0 y 100. El usuario debe
 *     adivinarlo y tiene para ello 5 oportunidades. Después de cada intento fallido, el
 *     programa dirá cuántas oportunidades quedan y si el número introducido es menor o
 *     mayor que el que ha pensado.
 *
 * @author Luis José Sánchez
 */
public class S06Ejercicio14 {

  public static void main(String[] args) {

    int oportunidades = 5;
    int numeroPensado;
    int minimo = 0;
    int maximo = 100;
    boolean acertado = false;
    int mayorMenorOIgual;

    System.out.println("Piensa un número del 0 al 100. Intentaré adivinarlo en 5 intentos.");
    System.out.println("Pulsa la tecla INTRO cuando estés preparado.");
    System.console().readLine();

    do {
      numeroPensado = (int)(Math.random() * (maximo - minimo) + minimo);
      System.out.println("¿Es el " + numeroPensado + "?");
      System.out.print("El número que estás pensando es 1) mayor 2) menor 3) el mismo: ");
      mayorMenorOIgual = Integer.parseInt(System.console().readLine());
      oportunidades--;

      if ( (mayorMenorOIgual == 1) && (oportunidades > 0) )
        minimo = numeroPensado + 1;

      if ( (mayorMenorOIgual == 2) && (oportunidades > 0) )
        maximo = numeroPensado - 1;

      if (mayorMenorOIgual == 3) {
        acertado = true;
        System.out.println("¡Bien! ¡he acertado!");
      }
```

```
    } while(!acertado && (oportunidades > 0));

    if (!acertado) {
      System.out.println("Vaya, no he conseguido acertar el número.");
    }
  }
}
```

Ejercicio 15

Fichero: S06Ejercicio15.java

```
/**
 * 6. Números aleatorios
 *
 * Solución al ejercicio 15.
 *
 * "Aprende Java con Ejercicios" (https://leanpub.com/aprendejava)
 *
 * @author Luis José Sánchez
 */
public class S06Ejercicio15 {

  public static void main(String[] args) {

    int notas = 4 * (int)(Math.random() * 7 + 1);
    String nota = "";
    String primeraNota = "";

    for (int contadorNota = 1; contadorNota <= notas; contadorNota++) {
      switch((int)(Math.random() * 7)) {
        case 0:
          nota = "do";
          break;
        case 1:
          nota = "re";
          break;
        case 2:
          nota = "mi";
          break;
        case 3:
          nota = "fa";
          break;
        case 4:
          nota = "sol";
```

```
          break;
        case 5:
          nota = "la";
          break;
        case 6:
          nota = "si";
          break;
        default:
      }

      if (contadorNota == 1) {
        primeraNota = nota;
      }

      if (contadorNota == notas) {
        nota = primeraNota;
      }

      System.out.print(nota + " ");

      if ( contadorNota == notas ) {
        System.out.print("||");
      } else if ( contadorNota % 4 == 0 ) {
        System.out.print("| ");
      }

    } // for
  }
}
```

Ejercicio 16

Fichero: S06Ejercicio16.java

```
/**
 * Aprende Java con Ejercicios (https://leanpub.com/aprendejava)
 *
 * Capítulo 6. Números aleatorios.
 *
 * Ejercicio 16
 *
 * @author Luis José Sánchez
 */
public class S06Ejercicio16 {
  public static void main(String[] args) {
```

```
int figura;
int figura1 = 0;
int figura2 = 0;
int figura3 = 0;

for (int i = 0; i < 3; i++) {
  figura = (int)(5 * Math.random());

  switch(figura) {
    case 0:
      System.out.print("corazón ");
      break;
    case 1:
      System.out.print("diamante ");
      break;
    case 2:
      System.out.print("herradura ");
      break;
    case 3:
      System.out.print("campana ");
      break;
    case 4:
      System.out.print("limón ");
      break;
    default:
  }

  switch(i) {
    case 0:
      figura1 = figura;
      break;
    case 1:
      figura2 = figura;
    break;
    case 2:
      figura3 = figura;
    break;
    default:
  }
}

if ((figura1 != figura2) && (figura2 != figura3) && (figura1 != figura3)) {
  System.out.println("\nLo siento, ha perdido.");
} else if ((figura1 == figura2) && (figura2 == figura3)) {
  System.out.println("\nEnhorabuena, ha ganado 10 monedas.");
```

```
    } else {
      System.out.println("\nBien, ha recuperado su moneda.");
    }
  }
}
```

Ejercicio 17

Fichero: S06Ejercicio17.java

```
/**
 * Aprende Java con Ejercicios (https://leanpub.com/aprendejava)
 *
 * Capítulo 6. Números aleatorios.
 *
 * Ejercicio 17
 *
 * @author Luis José Sánchez
 */
public class S06Ejercicio17 {
  public static void main(String[] args) {

    System.out.print("Por favor, introduzca la altura de la pecera (como mínimo 4): ");
    int alto = Integer.parseInt(System.console().readLine());

    System.out.print("Ahora introduzca la anchura (como mínimo 4): ");
    int ancho = Integer.parseInt(System.console().readLine());

    int posicion = 0;

    int posicionPez = (int)(Math.random()*(alto - 2)*(ancho - 2));

    // Pinta la parte superior ////////////////////////////////////////////////
    for(int i = 0; i < ancho; i++) {
      System.out.print("*");
    }
    System.out.println();

    // Pinta la parte central /////////////////////////////////////////////////
    for(int i = 2; i < alto; i++) {
      System.out.print("*"); // parte izquierda de la pecera
      for(int j = 2; j < ancho; j++) {
        if (posicion == posicionPez) {
          System.out.print("&");
        } else {
```

```
          System.out.print(" ");
        }
        posicion++;
      } // for j
      System.out.println("*"); // parte derecha de la pecera
    } // for i

    // Pinta la parte inferior //////////////////////////////////////////////////
    for(int i = 0; i < ancho; i++) {
      System.out.print("*");
    }
  }
}
```

Ejercicio 18

Fichero: S06Ejercicio18.java

```
/**
 * Aprende Java con Ejercicios (https://leanpub.com/aprendejava)
 *
 * Capítulo 6. Números aleatorios.
 *
 * Ejercicio 18
 *
 * @author Luis José Sánchez
 */
public class S06Ejercicio18 {
  public static void main(String[] args) {

    // Todavía no se han visto arrays y por eso el ejercicio está
    // implementado exclusivamente con variables. Con arrays podría
    // ser mucho más corto.

    int c1;
    int c2;
    int c3;

    do {
      c1 = (int)(Math.random() * 6);
      c2 = (int)(Math.random() * 6);
      c3 = (int)(Math.random() * 6);
    } while ((c1 == c2) || (c1 == c3) || (c2 == c3));
```

```
String color1 = "";

switch(c1) {
  case 0:
    color1 = "rojo";
    break;
  case 1:
    color1 = "azul";
    break;
  case 2:
    color1 = "verde";
    break;
  case 3:
    color1 = "amarillo";
    break;
  case 4:
    color1 = "violeta";
    break;
  case 5:
    color1 = "naranja";
    break;
  default:
}

String color2 = "";

switch(c2) {
  case 0:
    color2 = "rojo";
    break;
  case 1:
    color2 = "azul";
    break;
  case 2:
    color2 = "verde";
    break;
  case 3:
    color2 = "amarillo";
    break;
  case 4:
    color2 = "violeta";
    break;
  case 5:
    color2 = "naranja";
    break;
  default:
```

```
    }

    String color3 = "";

    switch(c3) {
      case 0:
        color3 = "rojo";
        break;
      case 1:
        color3 = "azul";
        break;
      case 2:
        color3 = "verde";
        break;
      case 3:
        color3 = "amarillo";
        break;
      case 4:
        color3 = "violeta";
        break;
      case 5:
        color3 = "naranja";
        break;
      default:
    }

    System.out.print(color1 + ", " + color2 + ", " + color3);
  }
}
```

Ejercicio 18

Fichero: S06Ejercicio18Alternativo.java

```
/**
 * Aprende Java con Ejercicios (https://leanpub.com/aprendejava)
 *
 * Capítulo 6. Números aleatorios.
 *
 * Ejercicio 18
 *
 * @author Luis José Sánchez
 */
public class S06Ejercicio18Alternativo {
  public static void main(String[] args) {
```

```
// Todavía no se han visto arrays y por eso el ejercicio está
// implementado exclusivamente con variables. Con arrays podría
// ser mucho más corto.

int c1;
int c2;
int c3;

String color1 = "";
String color2 = "";
String color3 = "";

do {
  c1 = (int)(Math.random() * 6);
  c2 = (int)(Math.random() * 6);
  c3 = (int)(Math.random() * 6);
} while ((c1 == c2) || (c1 == c3) || (c2 == c3));

for (int i = 0; i < 3; i++) {
  String color = "";
  int c = 0;

  switch(i) {
    case 0:
      c = c1;
      break;
    case 1:
      c = c2;
      break;
    case 2:
      c = c3;
      break;
    default:
  }

  switch(c) {
    case 0:
      color = "rojo";
      break;
    case 1:
      color = "azul";
      break;
    case 2:
      color = "verde";
      break;
```

```
        case 3:
          color = "amarillo";
          break;
        case 4:
          color = "violeta";
          break;
        case 5:
          color = "naranja";
          break;
        default:
      }

      switch(i) {
        case 0:
          color1 = color;
          break;
        case 1:
          color2 = color;
          break;
        case 2:
          color3 = color;
          break;
        default:
      }
    }

    System.out.print(color1 + ", " + color2 + ", " + color3);
  }
}
```

Ejercicio 19

Fichero: S06Ejercicio19.java

```
/**
 * Aprende Java con Ejercicios (https://leanpub.com/aprendejava)
 *
 * Capítulo 6. Números aleatorios.
 *
 * Ejercicio 19
 *
 * @author Luis José Sánchez
 */
public class S06Ejercicio19 {
  public static void main(String[] args) {
```

```
    int numero;
    int maximoPar = -100;
    int minimoImpar = 200;
    int suma = 0;

    for (int i = 0; i < 50; i++) {
      numero = (int)(Math.random() * 301) - 100;

      System.out.print(numero + " ");

      if (numero % 2 == 0) { // el número es par
        if (numero > maximoPar) maximoPar = numero;
      } else { // el número es impar
        if (numero < minimoImpar) minimoImpar = numero;
      }

      suma += numero;
    }

    System.out.println("\nMáximo de los pares: " + maximoPar);
    System.out.println("Mínimo de los impares: " + minimoImpar);
    System.out.println("Media: " + suma / 50);
  }
}
```

Ejercicio 20

Fichero: S06Ejercicio20.java

```
/**
 * Aprende Java con Ejercicios (https://leanpub.com/aprendejava)
 *
 * Capítulo 6. Números aleatorios.
 *
 * Ejercicio 20
 *
 * @author Luis José Sánchez
 */
public class S06Ejercicio20 {
  public static void main(String[] args) {

    System.out.print("Por favor, indique la capacidad de la cuba en litros: ");
    int capacidad = Integer.parseInt(System.console().readLine());
```

```
    // Rellena la cuba con unos litros aleatorios
    // teniendo en cuenta no pasarse de la capacidad.
    int litros = (int)(Math.random() * (capacidad + 1));

    for (int i = 0; i < capacidad; i++) {
      if (i < (capacidad - litros)) {
        System.out.println("#    #");
      } else {
        System.out.println("#====#");
      }
    }

    System.out.println("######"); // fondo de la cuba
    System.out.print("La cuba tiene una capacidad de " + capacidad);
    System.out.print(" litros y contiene " + litros + " litros de agua.");
  }
}
```

Ejercicio 21

Fichero: S06Ejercicio21.java

```
/**
 * Aprende Java con Ejercicios (https://leanpub.com/aprendejava)
 *
 * Capítulo 6. Números aleatorios.
 *
 * Ejercicio 21
 *
 * @author Luis José Sánchez
 */
public class S06Ejercicio21 {
  public static void main(String[] args) {

    // Todavía no se han visto los arrays y por eso el ejercicio está
    // implementado exclusivamente con variables. Con arrays podría
    // ser mucho más corto.

    String moneda = "";
    String posicion = "";

    for (int i = 0; i < 5; i++) {
      switch((int)(Math.random() * 8)) {
        case 0:
          moneda = "1 céntimo";
```

```
          break;
        case 1:
          moneda = "2 céntimos";
          break;
        case 2:
          moneda = "5 céntimos";
          break;
        case 3:
          moneda = "10 céntimos";
          break;
        case 4:
          moneda = "20 céntimos";
          break;
        case 5:
          moneda = "50 céntimos";
          break;
        case 6:
          moneda = "1 euro";
          break;
        case 7:
          moneda = "2 euros";
        default:
      }

      switch((int)(Math.random() * 2)) {
        case 0:
          posicion = "cara";
          break;
        case 1:
          posicion = "cruz";
          break;
        default:
      }

      System.out.println(moneda + " - " + posicion);
    }
  }
}
```

Ejercicio 22

Fichero: S06Ejercicio22.java

```
/**
 * Aprende Java con Ejercicios (https://leanpub.com/aprendejava)
 *
 * Capítulo 6. Números aleatorios.
 *
 * Ejercicio 22
 *
 * @author Luis José Sánchez
 */
public class S06Ejercicio22 {
  public static void main(String[] args) {

    System.out.print("Por favor, introduzca la longitud de la serpiente en ");
    System.out.print("caracteres contando la cabeza: ");
    int longitud = Integer.parseInt(System.console().readLine());

    // pinta la cabeza
    System.out.println("            @");

    // pinta el cuerpo
    int i;
    int x = 13;

    while (longitud > 1) {
      // suma -1, 0 o 1 a la variable x
      x += (int)(Math.random() * 3) - 1;

      // pinta x - 1 espacios
      for (i = 1; i < x; i++) {
        System.out.print(" ");
      }

      // pinta el cuerpo
      System.out.println("#");

      longitud--;
    }
  }
}
```

Ejercicio 23

Fichero: S06Ejercicio23.java

```java
/**
 * Aprende Java con Ejercicios (https://leanpub.com/aprendejava)
 *
 * Capítulo 6. Números aleatorios.
 *
 * Ejercicio 23
 *
 * @author Luis José Sánchez
 */
public class S06Ejercicio23 {
  public static void main(String[] args) {

    String dado = "";

    for (int i = 0; i < 5; i++) {
      switch((int)(Math.random() * 6)) {
        case 0:
          dado = "As";
          break;
        case 1:
          dado = "K";
          break;
        case 2:
          dado = "Q";
          break;
        case 3:
          dado = "J";
          break;
        case 4:
          dado = "7";
          break;
        case 5:
          dado = "8";
          break;
        default:
      }
      System.out.print(dado + " ");
    }
  }
}
```

Ejercicio 24

Fichero: S06Ejercicio24.java

```java
/**
 * Aprende Java con Ejercicios (https://leanpub.com/aprendejava)
 *
 * Capítulo 6. Números aleatorios.
 *
 * Ejercicio 24
 *
 * @author Luis José Sánchez
 */
public class S06Ejercicio24 {
  public static void main(String[] args) {

    System.out.print("Por favor, introduzca un número entero positivo: ");
    long numeroIntroducido = Long.parseLong(System.console().readLine());

    long numero = numeroIntroducido;

    // calcula la longitud del número
    int longitud = 0;

    do {
      numero /= 10;
      longitud++;
    } while (numero > 0);

    // elige una posición al azar dentro del número
    int posicion = (int)(Math.random() * longitud) + 1;

    // extrae el dígito de esa posición
    System.out.println((numeroIntroducido / (long)(Math.pow(10, longitud - posicion))) % 10);
  }
}
```

Ejercicio 25

Fichero: S06Ejercicio25.java

```
/**
 * Aprende Java con Ejercicios (https://leanpub.com/aprendejava)
 *
 * Capítulo 6. Números aleatorios.
 *
 * Ejercicio 25
 *
 * @author Luis José Sánchez
 */
public class S06Ejercicio25 {
  public static void main(String[] args) {

    for (int i = 0; i < 100; i++) {
      int numero = (int)(Math.random() * 191) + 10;

      // Comprueba si el número es primo
      boolean esPrimo = true;
      for (int j = 2; j < numero; j++) {
        if ((numero % j) == 0) {
          esPrimo = false;
        }
      }

      if (esPrimo) {
        System.out.print(" #" + numero + "# ");
      } else if ((numero % 5) == 0) {// múltiplo de 5
        System.out.print(" [" + numero + "] ");
      } else {
        System.out.print(" " + numero + " ");
      }
    } // for
  }
}
```

Ejercicio 26

Fichero: S06Ejercicio26.java

```
import java.util.Scanner;

public class S06Ejercicio26 {

  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);

    System.out.print("Introduzca la anchura de la tableta: ");
    int anchura = Integer.parseInt(s.nextLine());

    System.out.print("Introduzca la altura de la tableta: ");
    int altura = Integer.parseInt(s.nextLine());

    int posicionMordisco = (int) (Math.random() * (anchura * 2 + (altura - 2) * 2));

    int posicion = 0;
    for (int i = 0; i < altura; i++) {
      for (int j = 0; j < anchura; j++) {
        boolean estaEnBorde = (i == 0) || (i == altura - 1) || (j == 0) || (j == anchura - 1);

        if ((posicion == posicionMordisco) && estaEnBorde) {
          System.out.print(" ");
        } else {
          System.out.print("*");
        }

        if (estaEnBorde) {
          posicion++;
        }
      }
      System.out.println();
    }
  }

}
```

Ejercicio 27

Fichero: S06Ejercicio27.java

```
import java.util.Scanner;

public class S06Ejercicio27 {

  public static void main(String[] args) {

    Scanner s = new Scanner(System.in);

    System.out.print("Turno del jugador (introduzca piedra, papel o tijera): ");
    String jugador = s.nextLine();

    int mano = (int) (Math.random() * 3);
    String ordenador = "";

    switch (mano) {
      case 0:
        ordenador = "piedra";
        break;
      case 1:
        ordenador = "papel";
        break;
      case 2:
        ordenador = "tijera";
        break;
      default:
    }

    System.out.println("Turno del ordenador: " + ordenador);

    if (jugador.equals(ordenador)) {
      System.out.println("Empate");
    } else {
      int ganador = 2;
      switch (jugador) {
        case "piedra":
          if (ordenador.equals("tijera")) {
            ganador = 1;
          }
          break;
        case "papel":
          if (ordenador.equals("piedra")) {
            ganador = 1;
          }
          break;
        case "tijera":
          if (ordenador.equals("papel")) {
```

```
          ganador = 1;
        }
        break;
      default:

    }

    if (ganador == 1) {
      System.out.println("Gana el jugador");
    } else {
      System.out.println("Gana el ordenador");
    }

  }
}

}
```

Ejercicio 28

Fichero: S06Ejercicio28.java

```
import java.util.Scanner;

public class S06Ejercicio28 {

  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    System.out.print("Introduzca el tamaño del array: ");
    int n = Integer.parseInt(s.nextLine());

    int[] a = new int[n];

    for (int i = 0; i < n; i++) {
      a[i] = (int) (Math.random() * 201);
    }

    System.out.println("\nArray original:");

    System.out.print("Índice ");
    for (int i = 0; i < a.length; i++) {
      System.out.printf("%5d", i);
    }

    System.out.print("\nValor  ");
```

```
    for (int i = 0; i < a.length; i++) {
      System.out.printf("%5d", a[i]);
    }

    int[] r = new int[n];
    int izquierda = 0;
    int derecha = n - 1;

    for (int i = 0; i < n; i++) {
      if (i % 2 == 0) {
        r[izquierda++] = a[i];
      } else {
        r[derecha--] = a[i];
      }
    }

    System.out.println("\n\nArray resultado:");

    System.out.print("Índice ");
    for (int i = 0; i < r.length; i++) {
      System.out.printf("%5d", i);
    }

    System.out.print("\nValor  ");
    for (int i = 0; i < r.length; i++) {
      System.out.printf("%5d", r[i]);
    }
  }

}
```

Ejercicio 29

Fichero: S06Ejercicio29.java

```
import java.util.Scanner;

public class S06Ejercicio29 {

  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);

    System.out.println("1. Primavera");
    System.out.println("2. Verano");
    System.out.println("3. Otoño");
```

```
  System.out.println("4. Invierno");
  System.out.print("Seleccione la estación del año (1-4): ");
  int estacion = Integer.parseInt(s.nextLine());

  int temperaturaMinima = 0;
  int temperaturaMaxima = 0;
  String cielo = "";

  switch(estacion) {
    case 1: // Primavera
      temperaturaMinima = (int)(Math.random() * 16 + 15);
      temperaturaMaxima = (int)(Math.random() * 16 + 15);
      cielo = Math.random() <= 0.6 ? "Soleado" : "Nublado";
      break;
    case 2: // Verano
      temperaturaMinima = (int)(Math.random() * 21 + 25);
      temperaturaMaxima = (int)(Math.random() * 21 + 25);
      cielo = Math.random() <= 0.8 ? "Soleado" : "Nublado";
      break;
    case 3: // Otoño
      temperaturaMinima = (int)(Math.random() * 11 + 20);
      temperaturaMaxima = (int)(Math.random() * 11 + 20);
      cielo = Math.random() <= 0.4 ? "Soleado" : "Nublado";
      break;
    case 4: // Invierno
      temperaturaMinima = (int)(Math.random() * 26);
      temperaturaMaxima = (int)(Math.random() * 26);
      cielo = Math.random() <= 0.2 ? "Soleado" : "Nublado";
      break;
    default:
      System.out.println("La estación seleccionada no es correcta.");
  }

  // Si la mínima es mayor, se intercambian las temperaturas
  if (temperaturaMinima > temperaturaMaxima) {
    int aux = temperaturaMinima;
    temperaturaMinima = temperaturaMaxima;
    temperaturaMaxima = aux;
  }

  System.out.println("Previsión del tiempo para mañana");
  System.out.println("--------------------------------");
  System.out.println("Temperatura mínima: " + temperaturaMinima);
  System.out.println("Temperatura máxima: " + temperaturaMaxima);
  System.out.println(cielo);
}
```

```
}
```

Ejercicio 30

Fichero: S06Ejercicio30.java

```
import java.util.Scanner;

public class S06Ejercicio30 {

  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);

    System.out.print("Por favor, introduzca la altura de la pecera (como mínimo 4): ");
    int alto = Integer.parseInt(s.nextLine());

    System.out.print("Ahora introduzca la anchura (como mínimo 4): ");
    int ancho = Integer.parseInt(s.nextLine());

    int posicion = 0;
    int posicionPez;
    int posicionCaballito;
    int posicionCaracola;

    do {
      posicionPez = (int) (Math.random() * (alto - 2) * (ancho - 2));
      posicionCaballito = (int) (Math.random() * (alto - 2) * (ancho - 2));
      posicionCaracola = (int) (Math.random() * (alto - 2) * (ancho - 2));
    } while (
      (posicionPez == posicionCaballito) ||
      (posicionPez == posicionCaracola) ||
      (posicionCaballito == posicionCaracola));

    for (int i = 0; i < ancho; i++) {
      System.out.print("*");
    }
    System.out.println();

    for (int i = 2; i < alto; i++) {
      System.out.print("*");
      for (int j = 2; j < ancho; j++) {
        if (posicion == posicionPez) {
          System.out.print("&");
        } else if (posicion == posicionCaracola) {
```

```
          System.out.print("@");
        } else if (posicion == posicionCaballito) {
          System.out.print("$");
        } else {
          System.out.print(" ");
        }
        posicion++;
      }
      System.out.println("*");
    }

    for (int i = 0; i < ancho; i++) {
      System.out.print("*");
    }
    System.out.println();
  }

}
```

Ejercicio 31

Fichero: S06Ejercicio31.java

```
import java.util.Scanner;

public class S06Ejercicio31 {

  public static void main(String[] args) {
    System.out.println("□□□□□□□□□□□□□□□□□□□□□□□□□");
    System.out.println("□ □ C □ R □ A □ P □ S □ □");
    System.out.println("□□□□□□□□□□□□□□□□□□□□□□□□□");

    Scanner s = new Scanner(System.in);
    System.out.print("¿Cuánto dinero quiere apostar? ");
    int dinero = Integer.parseInt(s.nextLine());

    boolean juegoTerminado = false;

    int dado1 = (int) (Math.random() * 6 + 1);
    int dado2 = (int) (Math.random() * 6 + 1);
    int suma = dado1 + dado2;

    System.out.println("Dado 1: " + dado1);
    System.out.println("Dado 2: " + dado2);
    System.out.println("Suma: " + suma);
```

```
  switch (suma) {
    case 7:
    case 11:
      System.out.println("¡Enhorabuena! ¡Ha ganado otros " + dinero + " €!");
      System.out.println("¡Ahora tiene " + dinero * 2 + " €!");
      break;
    case 2:
    case 3:
    case 12:
      System.out.println("Lo siento, ha perdido todo su dinero □");
      break;
    default:
      System.out.print("Tiene que seguir tirando, debe conseguir el ");
      System.out.println(suma + " para ganar.");
      System.out.println("Si obtiene un 7, perderá la partida.");
      System.out.println("Pulse INTRO para tirar los dados.");
      s.nextLine();

      boolean partidaTerminada = false;

      do {
        dado1 = (int) (Math.random() * 6 + 1);
        dado2 = (int) (Math.random() * 6 + 1);

        System.out.println("Dado 1: " + dado1);
        System.out.println("Dado 2: " + dado2);
        System.out.println("Suma: " + (dado1 + dado2));

        if ((dado1 + dado2) == suma) {
          System.out.println("¡Enhorabuena! ¡Ha ganado otros " + dinero + " €!");
          System.out.println("¡Ahora tiene " + dinero * 2 + " €!");
          partidaTerminada = true;
        } else if ((dado1 + dado2) == 7) {
          System.out.println("Lo siento, ha perdido todo su dinero □");
          partidaTerminada = true;
        } else {
          System.out.println("Continúe jugando.");
          System.out.println("Pulse INTRO para tirar los dados.");
          s.nextLine();
        }
      } while (!partidaTerminada);
  }

}
```

```
}
```

Ejercicio 32

Fichero: S06Ejercicio32.java

```
import java.util.Scanner;

public class S06Ejercicio32 {
  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);

    System.out.print("Introduzca la longitud del sendero en metros: ");
    int longitudSendero = Integer.parseInt(s.nextLine());

    int espaciosPorDelante = 6;

    for (int i = 0; i < longitudSendero; i++) {
      // Espacios por delante
      for (int j = 0; j < espaciosPorDelante; j++) {
        System.out.print(' ');
      }

      // Borde izquierdo del sendero
      System.out.print('|');

      // Parte interior del sendero
      int posicionObstaculo = -1;
      char obstaculo = '*'; // planta por defecto

      if ((int) (Math.random() * 2) == 0) { // hay obstáculo
        posicionObstaculo = (int) (Math.random() * 4);
        if ((int) (Math.random() * 2) == 0) { // piedra
          obstaculo = 'O';
        }
      }

      for (int j = 0; j < 4; j++) {
        System.out.print(j == posicionObstaculo ? obstaculo : ' ');
      }

      // Borde derecho del sendero
      System.out.println('|');

      espaciosPorDelante += (int) (Math.random() * 3) - 1;
```

```
    }
  }

}
```

Ejercicio 32 (alternativo)

Fichero: S06Ejercicio32alt.java

```
import java.util.Scanner;

public class S06Ejercicio32alt {
  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);

    System.out.print("Introduzca la longitud del sendero en metros: ");
    int longitudSendero = Integer.parseInt(s.nextLine());

    int espaciosPorDelante = 6;

    for (int i = 0; i < longitudSendero; i++) {
      // Espacios por delante
      for (int j = 0; j < espaciosPorDelante; j++) {
        System.out.print(' ');
      }

      // Borde izquierdo del sendero
      System.out.print("□");

      // Parte interior del sendero
      int posicionObstaculo = -1;
      String obstaculo = "□"; // planta por defecto

      if ((int) (Math.random() * 2) == 0) { // hay obstáculo
        posicionObstaculo = (int) (Math.random() * 4);
        if ((int) (Math.random() * 2) == 0) { // piedra
          obstaculo = "□";
        }
      }

      for (int j = 0; j < 4; j++) {
        System.out.print(j == posicionObstaculo ? obstaculo : ' ');
      }

      // Borde derecho del sendero
```

```
      System.out.println("□");

      espaciosPorDelante += (int) (Math.random() * 3) - 1;
    }
  }

}
```

Arrays

Arrays de una dimensión

Ejercicio 1

Fichero: S71Ejercicio01.java

```
/**
 * 7.1 Arrays
 *
 * 1. Define un array de 12 números enteros con nombre num y asigna los valores
 *    según la tabla que se muestra a continuación. Muestra el contenido de todos
 *    los elementos del array. ¿Qué sucede con los valores de los elementos que
 *    no han sido inicializados?
 *    Posición  0   1   2   3   4   5   6   7   8   9   10   11
 *    Valor    39  -2           0      14       5  120
 *
 * @author Luis José Sánchez
 */
public class S71Ejercicio01 {

  public static void main(String[] args) {

      int[] num = new int[12];

      num[0] = 39;
      num[1] = -2;
      num[4] = 0;
      num[6] = 14;
      num[8] = 5;
      num[9] = 120;

      for (int i = 0; i < 12; i++) {
        System.out.printf("num[%2d]: %d\n", i, num[i]);
      }

  // Se puede observar que los valores del array se inicializan automáticamente a 0.
  }
}
```

Ejercicio 2

Fichero: S71Ejercicio02.java

```
/**
 * 7.1 Arrays
 *
 * 2. Define un array de 10 simboloes con nombre "simbolo" y asigna
 *    valores a los elementos según la siguiente tabla:
 *    Posición  0   1   2   3   4   5   6   7   8   9
 *    Valor    'a' 'x'         '@'     '   ' '+' 'Q'
 *
 *    Muestra el contenido de todos los elementos del array. ¿Qué sucede
 *    con los valores de los elementos que no han sido inicializados?
 *
 * @author Luis José Sánchez
 */
public class S71Ejercicio02 {

  public static void main(String[] args) {

    char[] simbolo = new char[10];

    simbolo[0] = 'a';
    simbolo[1] = 'x';
    simbolo[4] = '@';
    simbolo[6] = ' ';
    simbolo[7] = '+';
    simbolo[8] = 'Q';

    for (int i = 0; i < 10; i++) {
      System.out.println("simbolo[" + i + "]: " + simbolo[i]);
    }
    // Se puede observar que los valores del array se inicializan automáticamente
    // al simbolo vacío.
  }
}
```

Ejercicio 3

Fichero: S71Ejercicio03.java

```
/**
 * 7.1 Arrays
 *
 * 3. Escribe un programa que lea 10 números por teclado y que luego
 *    los muestre en orden inverso, es decir, el primero que se
 *    introduce es el último en mostrarse y viceversa.
 *
 * @author Luis José Sánchez
 */
public class S71Ejercicio03 {

  public static void main(String[] args) {

    int[] n = new int[10];
    int i;

    System.out.println("Por favor, introduzca 10 números enteros.");
    System.out.println("Pulse la tecla INTRO después de introducir cada número.");

    for (i = 0; i < 10; i++) {
      n[i] = Integer.parseInt(System.console().readLine());
    }

    System.out.println("\nLos números introducidos, al revés, son los siguientes:");
    for (i = 9; i >= 0; i--) {
      System.out.println(n[i]);
    }
  }
}
```

Ejercicio 4

Fichero: S71Ejercicio04.java

```
/**
 * 7.1 Arrays
 *
 * 4. Define tres arrays de 20 números enteros cada una, con nombres
 *    “numero”, “cuadrado” y “cubo”. Carga el array “numero” con valores
 *    aleatorios entre 0 y 100. En el array “cuadrado” se deben almacenar
 *    los cuadrados de los valores que hay en el array “numero”. En el
 *    array “cubo” se deben almacenar los cubos de los valores que hay
 *    en “numero”. A continuación, muestra el contenido de los tres arrays
 *    dispuesto en tres columnas.
 *
```

```
 * @author Luis José Sánchez
 */

public class S71Ejercicio04 {

  public static void main(String[] args) {

      int[] numero = new int[20];
      int[] cuadrado = new int[20];
      int[] cubo = new int[20];

      int i;

      for (i = 0; i < 20; i++) {
        numero[i] = (int)(Math.random()*101);
        cuadrado[i] = numero[i] * numero[i];
        cubo[i] = cuadrado[i] * numero[i];
      }

      System.out.println("\nA continuación se muestran en tres columnas, un número aleatorio e\
ntre 0 y 100, su cuadrado y su cubo:\n");

      System.out.println("  n  │   n²  │     n³ \n─────┼───────┼─────────");
      for (i = 0; i < 20; i++) {
        System.out.printf("%4d │ %5d │%8d\n", numero[i], cuadrado[i], cubo[i]);
      }
  }
}
```

Ejercicio 5

Fichero: S71Ejercicio05.java

```
/**
 * Aprende Java con Ejercicios (https://leanpub.com/aprendejava)
 *
 * Capítulo 7. Arrays
 *
 * @author Luis José Sánchez
 */
public class S71Ejercicio05 {

  public static void main(String[] args) {

    int[] numero = new int[10];
```

```
    int maximo = Integer.MIN_VALUE;
    int minimo = Integer.MAX_VALUE;
    int i;

    System.out.println("Vaya introduciendo números enteros y pulsando INTRO:");

    for (i = 0; i < 10; i++) {
      numero[i] = Integer.parseInt(System.console().readLine());

      if (numero[i] < minimo) {
        minimo = numero[i];
      }

      if (numero[i] > maximo)  {
        maximo = numero[i];
      }
    }

    System.out.println();

    for (i = 0; i < 10; i++) {
      System.out.print(numero[i]);
      if (numero[i] == maximo) {
        System.out.print(" máximo");
      }

      if (numero[i] == minimo) {
        System.out.print(" mínimo");
      }
      System.out.println();
    }
  }
}
```

Ejercicio 6

Fichero: S71Ejercicio06.java

```java
/**
 * 7.1 Arrays
 *
 * 6. Escribe un programa que lea 15 números por teclado y que los almacene en un array. Rota \
los
 *    elementos de ese array, es decir, el elemento de la posición 0 debe pasar a la posición \
1,
 *    el de la 1 a la 2, etc. El número que se encuentra en la última posición debe pasar a la
 *    posición 0. Finalmente, muestra el contenido del array.
 *
 * @author Luis José Sánchez
 */
public class S71Ejercicio06 {

  public static void main(String[] args) {

      int[] numero = new int[15];
      int i;

      System.out.println("Vaya introduciendo números enteros y pulsando INTRO:");

      for (i = 0; i < 15; i++) {
        numero[i] = Integer.parseInt(System.console().readLine());
      }

      System.out.println();

      // Muestra el array original ///////////////
      System.out.println("Array original:");
      for (i = 0; i < 15; i++) {
        System.out.printf("|%3d ", i);
      }
      System.out.println("|");
      for (i = 0; i < 75; i++) {
        System.out.print("□");
      }
      System.out.println("□");
      for (i = 0; i < 15; i++) {
        System.out.printf("|%3d ", numero[i]);
      }
      System.out.println("|");
      ////////////////////////////////////////////

      // rota una posición a la derecha //////////
      int aux = numero[14];
      for (i = 14; i > 0; i--) {
```

```
        numero[i] = numero[i-1];
      }
      numero[0] = aux;
      ////////////////////////////////////////////

      // Muestra el array rotado /////////////////
      System.out.println("\nArray rotado a la derecha una posición:");
      for (i = 0; i < 15; i++) {
        System.out.printf("|%3d ", i);
      }
      System.out.println("|");
      for (i = 0; i < 75; i++) {
        System.out.print("□");
      }
      System.out.println("□");
      for (i = 0; i < 15; i++) {
        System.out.printf("|%3d ", numero[i]);
      }
      System.out.println("|");
      ////////////////////////////////////////////
  }
}
```

Ejercicio 7

Fichero: S71Ejercicio07.java

```
/**
 * 7.1 Arrays
 *
 * 7. Escribe un programa que genere 100 números aleatorios del 0 al 20 y que los muestre por
 *    pantalla separados por espacios. El programa pedirá entonces por teclado dos valores y a
 *    continuación cambiará todas las ocurrencias del primer valor por el segundo en la lista
 *    generada anteriormente. Los números que se han cambiado deben aparecer entrecomillados.
 *
 * @author Luis José Sánchez
 */
public class S71Ejercicio07 {

  public static void main(String[] args) {

    int[] numero = new int[100];
    int i;
    String verde = "\033[32m";
    String blanco = "\033[37m";
```

```
    // Rellena el array con números aleatorios
    for (i = 0; i < 100; i++) {
      numero[i] = (int)(Math.random() * 21);
      System.out.print(numero[i] + "  ");
    }

    System.out.print("\nIntroduzca un número de los que se han mostrado: ");
    int valor1 = Integer.parseInt(System.console().readLine());
    System.out.print("Introduzca el valor por el que quiere sustituirlo: ");
    int valor2 = Integer.parseInt(System.console().readLine());

    System.out.println();

    for (i = 0; i < 100; i++) {
      if (numero[i] == valor1) {
        numero[i] = valor2;
        System.out.print(verde + "\"" + numero[i] + "\"  ");
      } else {
        System.out.print(blanco + numero[i] + "  ");
      }
    }

    System.out.println();
  }
}
```

Ejercicio 8

Fichero: S71Ejercicio08.java

```
/**
 * 7.1 Arrays
 *
 * 8. Realiza un programa que pida la temperatura media que ha hecho en cada mes de un determi\
nado
 *    año y que muestre a continuación un diagrama de barras horizontales con esos datos. Las \
barras
 *    del diagrama se pueden dibujar a base de asteriscos o cualquier otro carácter.
 *
 * @author Luis José Sánchez
 */
public class S71Ejercicio08 {

  public static void main(String[] args) {
```

```
    String[] mes = {
      "enero", "febrero", "marzo", "abril", "mayo", "junio",
      "julio", "agosto", "septiembre", "octubre", "noviembre", "diciembre"
    };

    int[] temperatura = new int[12];
    int i, j;

    String verde = "\033[32m";
    String naranja = "\033[33m";
    String azul = "\033[34m";
    String morado = "\033[35m";
    String blanco = "\033[37m";

    for (i = 0; i < 12; i++) {
      System.out.print("Introduzca la temperatura media de " + mes[i] + ": ");
      temperatura[i] = Integer.parseInt(System.console().readLine());
    }

    for (i = 0; i < 12; i++) {
      System.out.printf(azul + "%12s " + verde + "|", mes[i]);
      for (j = 0; j < temperatura[i]; j++) {
        System.out.print(morado + "□");
      }
      System.out.println(naranja + " " + temperatura[i] + "ºC" + blanco);
    }
  }
}
```

Ejercicio 9

Fichero: S71Ejercicio09.java

```
/**
 * 7.1 Arrays
 *
 * 9. Realiza un programa que pida 8 números enteros y que luego muestre esos números
 *    junto con la palabra "par" o "impar" según proceda.
 *
 * @author Luis José Sánchez
 */
public class S71Ejercicio09 {

  public static void main(String[] args) {
```

```
    int[] numero = new int[8];
    int i;

    System.out.println("Introduzca 8 números enteros, pulse INTRO después de cada número:");

    for (i = 0; i < 8; i++) {
      numero[i] = Integer.parseInt(System.console().readLine());
    }

    for (i = 0; i < 8; i++) {
      if (numero[i] % 2 == 0) {
        System.out.println(numero[i] + " par");
      } else {
        System.out.println(numero[i] + " impar");
      }
    }
  }
}
```

Ejercicio 10

Fichero: S71Ejercicio10.java

```
/**
 * 7.1 Arrays
 *
 * 10. Escribe un programa que genere 20 números enteros aleatorios entre 0 y 100 y que los
 *     almacene en un array. El programa debe ser capaz de pasar todos los números pares a
 *     las primeras posiciones del array (del 0 en adelante) y todos los números impares a
 *     las celdas restantes. Utiliza arrays auxiliares si es necesario.
 *
 * @author Luis José Sánchez
 */
public class S71Ejercicio10 {

  public static void main(String[] args) {

    int[] n = new int[20];
    int[] par = new int[20];
    int[] impar = new int[20];
    int i;
    int pares = 0;
    int impares = 0;
```

```
    for (i = 0; i < 20; i++) {
      n[i] = (int)(Math.random() * 101);
      // Separa los números metiendo los pares en un array
      // y los impares en otro.
      if (n[i] % 2 == 0) {
        par[pares++] = n[i];
      } else {
        impar[impares++] = n[i];
      }
    }

    // Muestra el array original
    System.out.println("Array original:");
    for (i = 0; i < 20; i++) {
      System.out.print(n[i] + " ");
    }
    System.out.println();

    // Mete los pares en las primeras posiciones
    // del array original.
    for (i = 0; i < pares; i++) {
      n[i] = par[i];
    }

    // Mete los impares en los huecos que quedan.
    for (i = pares; i < 20; i++) {
      n[i] = impar[i - pares];
    }

    // Muestra el resultado.
    System.out.println("Array con los pares al principio:");
    for (i = 0; i < 20; i++) {
      System.out.print(n[i] + " ");
    }
  }
}
```

Ejercicio 10

Fichero: S71Ejercicio10alt.java

```
/**
 * 7.1 Arrays
 *
 * 10. Escribe un programa que genere 20 números enteros aleatorios entre 0 y 100 y que los
 *     almacene en un array. El programa debe ser capaz de pasar todos los números pares a
 *     las primeras posiciones del array (del 0 en adelante) y todos los números impares a
 *     las celdas restantes. Utiliza arrays auxiliares si es necesario.
 *
 *     --> Solución alternativa sin utilizar arrays auxiliares y que conserva el orden origina\
l.
 *
 * @author Luis José Sánchez
 */
public class S71Ejercicio10alt {

  public static void main(String[] args) {

    int[] n = new int[20];
    int i;
    int j;
    int k;
    int aux;

    // Rellena el array con números aleatorios entre 0 y 100
    System.out.println("Array original");
    for (i = 0; i < 20; i++) {
      n[i] = (int)(Math.random() * 101);
      System.out.print(n[i] + " ");
    }

    // En cada iteración del for se coloca un número par en su posición correcta
    for (i = 0; i < 20; i++) {

      // Busca el siguiente par a partir de la posición actual
      j = i;
      while ((n[j++] % 2 != 0) && (j < 20));

      // Desplaza el número par a su posición correcta (si quedan pares)
      if (j < 20) {
        for (k = j - 1; k > i; k--) {
          aux = n[k];
          n[k] = n[k - 1];
          n[k - 1] = aux;
        }
      }
    }
```

```
    // Muestra el resultado.
    System.out.println("\nArray con los pares al principio:");
    for (i = 0; i < 20; i++) {
      System.out.print(n[i] + " ");
    }
  }
}
```

Ejercicio 11

Fichero: S71Ejercicio11.java

```
/**
 * 7.1 Arrays
 *
 * 11. Realiza un programa que pida 10 números por teclado y que los almacene en un array.
 *     A continuación se mostrará el contenido de ese array junto al índice (0 - 9).
 *     Seguidamente el programa pasará los primos a las primeras posiciones, desplazando
 *     el resto de números (los que no son primos) de tal forma que no se pierda ninguno.
 *     Al final se debe mostrar el array resultante.
 *
 * @author Luis José Sánchez
 */
public class S71Ejercicio11 {
  public static void main(String[] args) {

    int[] n = new int[10];
    int[] primo = new int[10];
    int[] noPrimo = new int[10];
    int i;
    int j;
    int primos = 0;
    int noPrimos = 0;
    boolean esPrimo = false;

    System.out.println("Introduzca 10 números separados por INTRO:");

    for (i = 0; i < 10; i++) {
      n[i] = Integer.parseInt(System.console().readLine());

      // Comprueba si el número es o no primo.
      esPrimo = true;
      for (j = 2; j < n[i] - 1; j++) {
        if (n[i] % j == 0) {
```

```
          esPrimo = false;
        }
      }

      // Si el número es primo, se mete en un array y si
      // no es primo, se mete en otro diferente.
      if (esPrimo) {
        primo[primos++] = n[i];
      } else {
        noPrimo[noPrimos++] = n[i];
      }
    }

    // Muestra el array original
    System.out.println("\n\nArray original:");
    System.out.println("\n┌────────┬─────┬─────┬─────┬─────┬─────┬─────┬─────┬─────┬─────┬────\
┐");
    System.out.print("| Índice ");
    for (i = 0; i < 10; i++) {
      System.out.printf("|%4d ", i);
    }
    System.out.println("|\n├────────┼─────┼─────┼─────┼─────┼─────┼─────┼─────┼─────┼─────┼────\
─┤");
    System.out.print("| Valor  ");
    for (i = 0; i < 10; i++) {
      System.out.printf("|%4d ", n[i]);
    }
    System.out.println("|\n└────────┴─────┴─────┴─────┴─────┴─────┴─────┴─────┴─────┴─────┴────\
─┘");

    // Los números primos se menten en las primeras
    // posiciones del array original.
    for (i = 0; i < primos; i++) {
      n[i] = primo[i];
    }

    // Los números que no son primos se colocan al final.
    for (i = primos; i < primos + noPrimos; i++) {
      n[i] = noPrimo[i - primos];
    }

    // Muestra el resultado.
    System.out.println("\n\nArray con los primos al principio:");
    System.out.println("\n┌────────┬─────┬─────┬─────┬─────┬─────┬─────┬─────┬─────┬─────┬────\
┐");
    System.out.print("| Índice ");
```

```
    for (i = 0; i < 10; i++) {
      System.out.printf("|%4d ", i);
    }
    System.out.println("|\n├─────────┼──────┼──────┼──────┼──────┼──────┼──────┼──────┼──────┼──────┼───\
──┤");
    System.out.print("| Valor  ");
    for (i = 0; i < 10; i++) {
      System.out.printf("|%4d ", n[i]);
    }
    System.out.println("|\n└─────────┴──────┴──────┴──────┴──────┴──────┴──────┴──────┴──────┴──────┴───\
──┘");
  }
}
```

Ejercicio 12

Fichero: S71Ejercicio12.java

```
/**
 * Aprende Java con Ejercicios (https://leanpub.com/aprendejava)
 *
 * Capítulo 7. Arrays
 *
 * Ejercicio 12
 *
 * @author Luis José Sánchez
 */
public class S71Ejercicio12 {

  public static void main(String[] args) {

    int[] n = new int[10];
    int[] resultado = new int[10];
    int i;
    int nInicial;
    int nFinal;
    boolean valido;

    System.out.println("Introduzca 10 números separados por INTRO:");

    for (i = 0; i < 10; i++) {
      n[i] = Integer.parseInt(System.console().readLine());
    }

    // Muestra el array original.
```

```
    System.out.println("\n\nArray original:");
    System.out.println("\n┌────────┬─────┬─────┬─────┬─────┬─────┬─────┬─────┬─────┬─────┬────\
─┐");
    System.out.print("│ Índice ");
    for (i = 0; i < 10; i++) {
      System.out.printf("│%4d ", i);
    }
    System.out.println("│\n├────────┼─────┼─────┼─────┼─────┼─────┼─────┼─────┼─────┼─────┼────\
─┤");
    System.out.print("│ Valor  ");
    for (i = 0; i < 10; i++) {
      System.out.printf("│%4d ", n[i]);
    }
    System.out.println("│\n└────────┴─────┴─────┴─────┴─────┴─────┴─────┴─────┴─────┴─────┴────\
─┘");

    // Pide las posiciones inicial y final.
    do {
      valido = true;

      System.out.print("Introduzca la posición inicial (0 - 9): ");
      nInicial = Integer.parseInt(System.console().readLine());
      if ((nInicial < 0) || (nInicial > 9)) {
        System.out.println("Valor incorrecto, debe ser un número entre el 0 y el 9.");
        valido = false;
      }

      System.out.print("Introduzca la posición final (0 - 9): ");
      nFinal = Integer.parseInt(System.console().readLine());
      if ((nFinal < 0) || (nFinal > 9)) {
        System.out.println("Valor incorrecto, debe ser un número entre el 0 y el 9.");
        valido = false;
      }

      if (nInicial >= nFinal) {
        System.out.println("Valores incorrectos, la posición inicial debe ser menor que la pos\
ición final.");
        valido = false;
      }
    } while (!valido);

    // Muestra de nuevo el array original.
    System.out.println("\n\nArray original:");
    System.out.println("\n┌────────┬─────┬─────┬─────┬─────┬─────┬─────┬─────┬─────┬─────┬────\
─┐");
    System.out.print("│ Índice ");
```

```
    for (i = 0; i < 10; i++) {
      System.out.printf("|%4d ", i);
    }
    System.out.println("|\n├─────────┼──────┼──────┼──────┼──────┼──────┼──────┼──────┼──────┼──────┼────\
──┤");
    System.out.print("| Valor  ");
    for (i = 0; i < 10; i++) {
      System.out.printf("|%4d ", n[i]);
    }
    System.out.println("|\n└─────────┴──────┴──────┴──────┴──────┴──────┴──────┴──────┴──────┴──────┴────\
──┘");

    // Copia el array n en resultado.
    for (i = 0; i < 10; i++) {
      resultado[i] = n[i];
    }

    resultado[nFinal] = n[nInicial];

    for (i = nFinal + 1; i < 10; i++)
      resultado[i] = n[i - 1];

    resultado[0] = n[9];

    for (i = 0; i < nInicial; i++)
      resultado[i + 1] = n[i];

    // Muestra el resultado.
    System.out.println("\nArray resultante:");
    System.out.println("\n┌─────────┬──────┬──────┬──────┬──────┬──────┬──────┬──────┬──────┬──────┬─────\
┐");
    System.out.print("| Índice ");
    for (i = 0; i < 10; i++) {
      System.out.printf("|%4d ", i);
    }
    System.out.println("|\n├─────────┼──────┼──────┼──────┼──────┼──────┼──────┼──────┼──────┼──────┼────\
──┤");
    System.out.print("| Valor  ");
    for (i = 0; i < 10; i++) {
      System.out.printf("|%4d ", resultado[i]);
    }
    System.out.println("|\n└─────────┴──────┴──────┴──────┴──────┴──────┴──────┴──────┴──────┴──────┴────\
──┘");
  }
}
```

Ejercicio 13

Fichero: S71Ejercicio13.java

```
/**
 * Aprende Java con Ejercicios (https://leanpub.com/aprendejava)
 *
 * Capítulo 7. Arrays
 *
 * Ejercicio 13
 *
 * @author Luis José Sánchez
 */
public class S71Ejercicio13 {

  public static void main(String[] args) {

    int[] n = new int[100];
    int maximo = 0;
    int minimo = 100;

    // Genera los números y calcula el máximo y el mínimo
    for (int i = 0; i < 100; i++) {
      n[i] = (int)(Math.random() * 501);

      if (n[i] < minimo) {
        minimo = n[i];
      }

      if (n[i] > maximo)  {
        maximo = n[i];
      }
    }

    // Muestra el array original
    for (int elemento : n) {
      System.out.print(elemento + " ");
    }

    System.out.print("\n\n¿Qué quiere destacar? (1 - mínimo, 2 - máximo): ");
    int opcion = Integer.parseInt(System.console().readLine());

    int destacado; // número que se va a destacar del resto

    if (opcion == 1) {
      destacado = minimo;
```

```
    } else {
      destacado = maximo;
    }

    System.out.println();

    // Muestra el resultado.
    for (int elemento : n) {
      if (elemento == destacado) {
        System.out.print(" **" + elemento + "** ");
      } else {
        System.out.print(elemento + " ");
      }
    }
  }
}
```

Ejercicio 14

Fichero: S71Ejercicio14.java

```
/**
 * Aprende Java con Ejercicios (https://leanpub.com/aprendejava)
 *
 * Capítulo 7. Arrays
 *
 * Ejercicio 14
 *
 * @author Luis José Sánchez
 */
public class S71Ejercicio14 {
  public static void main(String[] args) {

    String[] color = {
      "verde", "rojo", "azul", "amarillo", "naranja", "rosa", "negro ", "blanco", "morado"
    };

    String[] palabra = new String[8];
    String[] resultado = new String[8];

    System.out.println("Introduzca 8 palabras (vaya pulsando [INTRO] entre una y otra.");

    int j = 0;

    for (int i = 0; i < 8; i++) {
```

```
      palabra[i] = System.console().readLine();

      // Si la palabra introducida es un color, la guarda en el array resultado.
      for (String c : color) {
        if (palabra[i].equals(c)) {
          resultado[j++] = c;
        }
      }
    }

    // Mete las palabras que no son colores al final del array resultado.
    for (int i = 0; i < 8; i++) {
      boolean esColor = false;

      for (String c : color) {
        if (palabra[i].equals(c)) {
          esColor = true;
        }
      }

      if (!esColor) {
        resultado[j++] = palabra[i];
      }
    }

    // Muestra el array original.
    System.out.println("\n\nArray original:");
    System.out.println("\n┌────────┬────────┬────────┬────────┬────────┬────────┬────────┬────\
────┐");
    for (int i = 0; i < 8; i++) {
      System.out.printf("|   %d    ", i);
    }
    System.out.println("|\n├────────┼────────┼────────┼────────┼────────┼────────┼────────┼────\
────┤");

    for (String p : palabra) {
      System.out.printf("|%-8s", p);
    }
    System.out.println("|\n└────────┴────────┴────────┴────────┴────────┴────────┴────────┴────\
────┘");

    // Muestra el array resultado.
    System.out.println("\n\nArray resultado:");
    System.out.println("\n┌────────┬────────┬────────┬────────┬────────┬────────┬────────┬────\
────┐");
    for (int i = 0; i < 8; i++) {
```

```
      System.out.printf("|   %d    ", i);
    }
    System.out.println("|\n├────────┼────────┼────────┼────────┼────────┼────────┼────────┼───\
─────┤");

    for (String r : resultado) {
      System.out.printf("|%-8s", r);
    }
    System.out.println("|\n└────────┴────────┴────────┴────────┴────────┴────────┴────────┴───\
─────┘");
  }
}
```

Ejercicio 15

Fichero: S71Ejercicio15.java

```
/**
 * Aprende Java con Ejercicios (https://leanpub.com/aprendejava)
 *
 * Capítulo 7. Arrays
 *
 * Ejercicio 15
 *
 * @author Luis José Sánchez
 */
public class S71Ejercicio15 {
  public static void main(String[] args) {

    int[] mesa = new int[10];

    // Rellena las mesas de forma aleatoria.
    for (int i = 0; i < 10; i++) {
      mesa[i] = (int)(Math.random() * 5);
    }

    int clientes; // número de clientes que llegan al restaurante buscando mesa

    do {
      // Muestra el estado de ocupación de las mesas
      System.out.println("\n┌─────────┬────┬────┬────┬────┬────┬────┬────┬────┬────┬────┐");
      System.out.print("|Mesa nº: ");
      for (int i = 1; i < 11; i++) {
        System.out.printf("| %2d ", i);
      }
```

```
      System.out.println("|\n├─────────┼────┼────┼────┼────┼────┼────┼────┼────┼────┼────┤");
      System.out.print("|Ocupación");
      for (int m : mesa) {
        System.out.printf("| %2d ", m);
      }
      System.out.println("|\n└─────────┴────┴────┴────┴────┴────┴────┴────┴────┴────┴────┘");

      System.out.print("¿Cuántos son? (Introduzca -1 para salir del programa): ");
      clientes = Integer.parseInt(System.console().readLine());

      if (clientes > 4) { // comprueba si el grupo de clientes es mayor a 4
        System.out.print("Lo siento, no admitimos grupos de 6, haga grupos de");
        System.out.println(" 4 personas como máximo e intente de nuevo.");
      } else {
        // Busca una mesa que esté vacía.
        int iVacia = 0;
        boolean hayMesaVacia = false;
        for(int i = 9; i >= 0; i--) {
          if (mesa[i] == 0) {
            iVacia = i;
            hayMesaVacia = true;
          }
        }

        if (hayMesaVacia) {
          mesa[iVacia] = clientes; // coloca a los clientes en la mesa libre
          System.out.println("Por favor, siéntense en la mesa número " + (iVacia + 1) + ".");
        } else {
          // Busca un hueco para todo el grupo.
          int iHueco = 0;
          boolean hayHueco = false;
          for(int i = 9; i >= 0; i--) {
            if (clientes <= (4 - mesa[i])) {
              iHueco = i;
              hayHueco = true;
            }
          }

          if (hayHueco) {
            mesa[iHueco] += clientes; // coloca a los clientes en el primer hueco disponible
            System.out.println("Tendrán que compartir mesa. Por favor, siéntense en la mesa nú\
mero " + (iHueco + 1) + ".");
          } else {
            System.out.println("Lo siento, en estos momentos no queda sitio.");
          }
        }
```

```
    }
  } while (clientes != -1);

  }
}
```

Ejercicio 16

Fichero: S71Ejercicio16.java

```
/**
 * Aprende Java con Ejercicios (https://leanpub.com/aprendejava)
 *
 * Capítulo 7. Arrays
 *
 * Ejercicio 16
 *
 * @author Luis José Sánchez
 */
public class S71Ejercicio16 {
  public static void main(String[] args) {

    int[] n = new int[20];

    for (int i = 0; i < 20; i++) {
      n[i] = (int)(Math.random() * 381) + 20;
      System.out.print(n[i] + " ");
    }

    System.out.print("\n\n¿Qué números quiere resaltar? ");
    System.out.print("(1 - los múltiplos de 5, 2 - los múltiplos de 7): ");
    int opcion = Integer.parseInt(System.console().readLine());

    int multiplo = (opcion == 1) ? 5 : 7;

    // Muestra el resultado.
    for (int elemento : n) {
      if (elemento % multiplo == 0) {
        System.out.print("[" + elemento + "] ");
      } else {
        System.out.print(elemento + " ");
      }
    }
  }
}
```

Ejercicio 17

Fichero: S71Ejercicio17.java

```
/**
 * Aprende Java con Ejercicios (https://leanpub.com/aprendejava)
 *
 * Capítulo 7. Arrays
 *
 * Ejercicio 17
 *
 * @author Luis José Sánchez
 */
public class S71Ejercicio17 {
  public static void main(String[] args) {

    int[] n = new int[10];

    // Genera el array
    for (int i = 0; i < 10; i++) {
      n[i] = (int)(Math.random() * 381) + 20;
    }

    // Muestra el array original.
    System.out.println("\n\nArray original:");
    System.out.println("\n┌────────┬─────┬─────┬─────┬─────┬─────┬─────┬─────┬─────┬─────┬─────\
┐");
    System.out.print("│ Índice ");
    for (int i = 0; i < 10; i++) {
      System.out.printf("│%4d ", i);
    }
    System.out.println("│\n├────────┼─────┼─────┼─────┼─────┼─────┼─────┼─────┼─────┼─────┼────\
─┤");
    System.out.print("│ Valor  ");
    for (int i = 0; i < 10; i++) {
      System.out.printf("│%4d ", n[i]);
    }
    System.out.println("│\n└────────┴─────┴─────┴─────┴─────┴─────┴─────┴─────┴─────┴─────┴────\
─┘");

    // Pide un número que esté dentro del array
    boolean existeNumero = false;
    int numeroIntroducido;
    do {
      System.out.print("\nIntroduzca uno de los números del array: ");
      numeroIntroducido = Integer.parseInt(System.console().readLine());
```

```
      // Comprueba si el número introducido se encuentra dentro del array
      for (int elemento : n) {
        if (elemento == numeroIntroducido) {
          existeNumero = true;
        }
      }

      if (!existeNumero) {
        System.out.println("Ese número no se encuentra en el array.");
      }
    } while (!existeNumero);

    // Rota los el array hasta que el número introducido se coloque en la posición 0
    while (n[0] != numeroIntroducido) {
      int aux = n[9];
      // Rotación a la derecha
      for (int i = 9; i > 0; i--) {
        n[i] = n[i -1];
      }
      n[0] = aux;
    }

    // Muestra el resultado.
    System.out.println("\nArray resultante:");
    System.out.println("\n┌────────┬─────┬─────┬─────┬─────┬─────┬─────┬─────┬─────┬─────┬────\
─┐");
    System.out.print("│ Índice ");
    for (int i = 0; i < 10; i++) {
      System.out.printf("│%4d ", i);
    }
    System.out.println("│\n├────────┼─────┼─────┼─────┼─────┼─────┼─────┼─────┼─────┼─────┼────\
─┤");
    System.out.print("│ Valor  ");
    for (int i = 0; i < 10; i++) {
      System.out.printf("│%4d ", n[i]);
    }
    System.out.println("│\n└────────┴─────┴─────┴─────┴─────┴─────┴─────┴─────┴─────┴─────┴────\
─┘");
  }
}
```

Ejercicio 18

Fichero: S71Ejercicio18.java

```
public class S71Ejercicio18 {

  public static void main(String[] args) {
    int[] a = new int[10];
    int[] menores = new int[10];
    int[] mayores = new int[10];
    int cuentaMenores = 0;
    int cuentaMayores = 0;

    // Genera el array original y clasifica los números
    for (int i = 0; i < 10; i++) {
      a[i] = (int) (Math.random() * 201);

      if (a[i] <= 100) {
        menores[cuentaMenores++] = a[i];
      } else {
        mayores[cuentaMayores++] = a[i];
      }
    }

    // Muestra el array original
    System.out.println("\nArray original:");
    System.out.println("┌────────┬─────┬─────┬─────┬─────┬─────┬─────┬─────┬─────┬─────┬─────┐\
");
    System.out.print("│ Índice ");
    for (int i = 0; i < 10; i++) {
      System.out.printf("│%4d ", i);
    }
    System.out.println("│");
    System.out.println("├────────┼─────┼─────┼─────┼─────┼─────┼─────┼─────┼─────┼─────┼─────┤\
");
    System.out.print("│ Valor  ");
    for (int i = 0; i < 10; i++) {
      System.out.printf("│%4d ", a[i]);
    }
    System.out.println("│");
    System.out.println("└────────┴─────┴─────┴─────┴─────┴─────┴─────┴─────┴─────┴─────┴─────┘\
");

    // Genera el array resultado
    int[] resultado = new int[10];
    int menoresColocados = 0;
    int mayoresColocados = 0;

    int j = 0;
    do {
```

```
      if (menoresColocados < cuentaMenores) {
        resultado[j++] = menores[menoresColocados++];
      }

      if (mayoresColocados < cuentaMayores) {
        resultado[j++] = mayores[mayoresColocados++];
      }

    } while (j < 10);


    // Muestra el resultado.
    System.out.println("\nArray resultado:");
    System.out.println("┌────────┬─────┬─────┬─────┬─────┬─────┬─────┬─────┬─────┬─────┬─────┐\
");
    System.out.print("| Índice ");
    for (int i = 0; i < 10; i++) {
      System.out.printf("|%4d ", i);
    }
    System.out.println("|");
    System.out.println("├────────┼─────┼─────┼─────┼─────┼─────┼─────┼─────┼─────┼─────┼─────┤\
");
    System.out.print("| Valor  ");
    for (int i = 0; i < 10; i++) {
      System.out.printf("|%4d ", resultado[i]);
    }
    System.out.println("|");
    System.out.println("└────────┴─────┴─────┴─────┴─────┴─────┴─────┴─────┴─────┴─────┴─────┘\
");
  }

}
```

Ejercicio 19

Fichero: S71Ejercicio19.java

```
import java.util.Scanner;

public class S71Ejercicio19 {

  public static void main(String[] args) {

    // Crea el array con valores aleatorios
    int[] n = new int[12];

    for (int i = 0; i < 12; i++) {
      n[i] = (int) (Math.random() * 201);
    }

    // Muestra el array original
    System.out.println("Array original:");

    System.out.print("\nÍndice ");

    for (int i = 0; i < 12; i++) {
      System.out.printf("%4d ", i);
    }

    System.out.print("\nValor  ");

    for (int i = 0; i < 12; i++) {
      System.out.printf("%4d ", n[i]);
    }

    // Pide el número y la posición donde colocarlo
    Scanner s = new Scanner(System.in);

    System.out.print("\n\nIntroduzca el número que quiere insertar: ");
    int numero = Integer.parseInt(s.nextLine());

    System.out.print("Introduzca la posición donde lo quiere insertar (0 - 11): ");
    int posicion = Integer.parseInt(s.nextLine());

    // Inserta el número y desplaza el resto hacia la derecha
    for (int i = 11; i > posicion; i--) {
      n[i] = n[i - 1];
    }

    n[posicion] = numero;

    // Muestra el resultado
    System.out.println("\nArray resultado:");
```

```
    System.out.print("\nÍndice ");

    for (int i = 0; i < 12; i++) {
      System.out.printf("%4d ", i);
    }

    System.out.print("\nValor  ");

    for (int i = 0; i < 12; i++) {
      System.out.printf("%4d ", n[i]);
    }
  }

}
```

Ejercicio 20

Fichero: S71Ejercicio20.java

```
import java.util.Scanner;

public class S71Ejercicio20 {

  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    System.out.print("Introduzca el número total de nombres de reyes: ");
    int n = Integer.parseInt(s.nextLine());

    String[] reyes = new String[n];

    System.out.println("Vaya introduciendo los nombres de los reyes y pulsando INTRO.");

    for (int i = 0; i < n; i++) {
      reyes[i] = s.nextLine();
    }

    for (int i = 0; i < n; i++) {
      int orden = 1;

      for (int j = 0; j < i; j++) {
        if (reyes[i].equals(reyes[j])) {
          orden++;
        }
      }
```

```
      System.out.println(reyes[i] + " " + orden + "º");
    }
  }

}
```

Ejercicio 21

Fichero: S71Ejercicio21.java

```
public class S71Ejercicio21 {

  public static void main(String[] args) {
    // Crea el array con valores aleatorios
    int[] n = new int[15];

    for (int i = 0; i < 15; i++) {
      n[i] = (int) (Math.random() * 501);
    }

    System.out.println("Array original:");

    for (int i = 0; i < 15; i++) {
      System.out.print(n[i] + "  ");
    }

    // Cincueriza el array
    for (int i = 0; i < 15; i++) {
      while (n[i] % 5 != 0) {
        n[i]++;
      };
    }

    // Muestra el resultado
    System.out.println("\n\nArray resultado:");

    for (int i = 0; i < 15; i++) {
      System.out.print(n[i] + "  ");
    }
  }

}
```

Arrays bidimensionales

Ejercicio 1

Fichero: S72Exercise01.java

```
/**
 * 7.2 Arrays bidimensionales
 *
 * 1. Define un array de números enteros de 3 filas por 6 columnas con
 *    nombre "num" y asigna los valores según la siguiente tabla:
 *    num:
 *           Columna 0   Columna 1   Columna 2   Columna 3   Columna 4   Columna 5
 *    Fila 0     0          30           2                                  5
 *    Fila 1    75                                              0
 *    Fila 2                            -2           9                     11
 *
 *    Muestra el contenido de todos los elementos del array dispuestos en
 *    forma de tabla como se muestra en la figura.
 *
 * @author Luis José Sánchez
 */
public class S72Exercise01 {
  public static void main(String[] args)
    throws InterruptedException { // Se añade esta línea para poder usar sleep

    int[][] num = new int[3][6]; // array de 3 filas por 6 columnas

    num[0][0] = 0;
    num[0][1] = 30;
    num[0][2] = 2;
    num[0][5] = 7;
    num[1][0] = 75;
    num[1][4] = 0;
    num[2][2] = -2;
    num[2][3] = 9;
    num[2][5] = 11;

    int fila;
    int columna;

    System.out.print("       ");
    for(columna = 0; columna < 6; columna++) {
      System.out.print("   Columna " + columna);
    }
    for(fila = 0; fila < 3; fila++) {
```

```
      System.out.print("\nFila " + fila);

      for(columna = 0; columna < 6; columna++) {
        System.out.printf("%9d   ", num[fila][columna]);
        Thread.sleep(500); // retardo de medio segundo
      }
    }
  }
}
```

Ejercicio 2

Fichero: S72Exercise02.java

```
/**
 * 7.2 Arrays bidimensionales
 *
 * 2. Escribe un programa que pida 20 números enteros.
 *    Estos números se deben introducir en un array de 4 filas por 5 columnas.
 *    El programa mostrará las sumas parciales de filas y columnas igual que
 *    si de una hoja de cálculo se tratara.
 *    La suma total debe aparecer en la esquina inferior derecha.
 *
 * @author Luis José Sánchez
 */
public class S72Exercise02 {
  public static void main(String[] args) {

    int[][] num = new int[4][5]; // array de 4 filas por 5 columnas

    int fila;
    int columna;

    // Lee los datos de teclado
    System.out.println("Por favor, introduzca los números (enteros) en el array");
    for(fila = 0; fila < 4; fila++) {
      for(columna = 0; columna < 5; columna++) {
        System.out.print("Fila " + fila + ", columna " + columna + ": ");
        num[fila][columna] = Integer.parseInt(System.console().readLine());
      }
    }

    // Muestra los datos y las sumas parciales de las filas
    int sumaFila;
```

```
    for(fila = 0; fila < 4; fila++) {
      sumaFila = 0;
      for(columna = 0; columna < 5; columna++) {
        System.out.printf("%7d   ", num[fila][columna]);
        sumaFila += num[fila][columna];
      }
      System.out.printf("|%7d\n", sumaFila);
    }

    // Muestra las sumas parciales de las columnas
    for(columna = 0; columna < 5; columna++) {
      System.out.print("----------");
    }
    System.out.println("-----------");

    int sumaColumna;
    int sumaTotal = 0;
    for(columna = 0; columna < 5; columna++) {
      sumaColumna = 0;
      for(fila = 0; fila < 4; fila++) {
        sumaColumna += num[fila][columna];
      }

      sumaTotal += sumaColumna;
      System.out.printf("%7d   ", sumaColumna);
    }
    System.out.printf("|%7d   ", sumaTotal);
  }
}
```

Ejercicio 3

Fichero: S72Exercise03.java

```
/**
 * 7.2 Arrays bidimensionales
 *
 * 3. Modifica el programa anterior de tal forma que los números que se introducen en
 *    el array se generen de forma aleatoria (valores entre 100 y 999).
 *
 * @author Luis José Sánchez
 */
public class S72Exercise03 {
  public static void main(String[] args) {
```

```
    int[][] num = new int[4][5]; // array de 4 filas por 5 columnas

    int fila;
    int columna;

    // Introduce valores aleatorios en el array
    for(fila = 0; fila < 4; fila++)
      for(columna = 0; columna < 5; columna++)
        num[fila][columna] = (int)(Math.random() * 900) + 100;

    // Muestra los datos y las sumas parciales de las filas
    int sumaFila;
    for(fila = 0; fila < 4; fila++) {
      sumaFila = 0;
      for(columna = 0; columna < 5; columna++) {
        System.out.printf("%7d   ", num[fila][columna]);
        sumaFila += num[fila][columna];
      }
      System.out.printf("|%7d\n", sumaFila);
    }

    // Muestra las sumas parciales de las columnas
    for(columna = 0; columna < 5; columna++) {
      System.out.print("----------");
    }
    System.out.println("-----------");

    int sumaColumna;
    int sumaTotal = 0;

    for(columna = 0; columna < 5; columna++) {
      sumaColumna = 0;
      for(fila = 0; fila < 4; fila++) {
        sumaColumna += num[fila][columna];
      }

      sumaTotal += sumaColumna;
      System.out.printf("%7d   ", sumaColumna);
    }
    System.out.printf("|%7d   ", sumaTotal);
  }
}
```

Ejercicio 4

Fichero: S72Exercise04.java

```java
/**
 * 7.2 Arrays bidimensionales
 *
 * 4. Modifica el programa anterior de tal forma que las sumas parciales y la suma total
 *    aparezcan en la pantalla con un pequeño retardo, dando la impresión de que el
 *    ordenador se queda pensando antes de mostrar los números.
 *
 * @author Luis José Sánchez
 */
public class S72Exercise04 {
  public static void main(String[] args)
    throws InterruptedException { // Se añade esta línea para poder usar sleep

    int[][] num = new int[4][5]; // array de 4 filas por 5 columnas

    int fila;
    int columna;

    // Introduce valores aleatorios en el array
    for(fila = 0; fila < 4; fila++) {
      for(columna = 0; columna < 5; columna++) {
        num[fila][columna] = (int)(Math.random() * 900) + 100;
      }
    }

    // Muestra los datos y las sumas parciales de las filas
    int sumaFila;
    for(fila = 0; fila < 4; fila++) {
      sumaFila = 0;
      for(columna = 0; columna < 5; columna++) {
        System.out.printf("%7d   ", num[fila][columna]);
        sumaFila += num[fila][columna];
        Thread.sleep(100);
      }
      System.out.printf("|%7d\n", sumaFila);
      Thread.sleep(500);
    }

    // Muestra las sumas parciales de las columnas
    for(columna = 0; columna < 5; columna++) {
      System.out.print("----------");
    }
    System.out.println("-----------");

    int sumaColumna;
    int sumaTotal = 0;
```

```
    for(columna = 0; columna < 5; columna++) {
      sumaColumna = 0;
      for(fila = 0; fila < 4; fila++) {
        sumaColumna += num[fila][columna];
      }

      sumaTotal += sumaColumna;
      System.out.printf("%7d   ", sumaColumna);
      Thread.sleep(500);
    }
    System.out.printf("|%7d   ", sumaTotal);
  }
}
```

Ejercicio 5

Fichero: S72Exercise05.java

```
/**
 * 7.2 Arrays bidimensionales
 *
 * 5. Realiza un programa que rellene un array de 6 filas por 10 columnas
 *    con números enteros positivos comprendidos entre 0 y 1000 (ambos incluidos).
 *    A continuación, el programa deberá dar la posición tanto del máximo como del
 *    mínimo.
 *
 * @author Luis José Sánchez
 */
public class S72Exercise05 {

  public static void main(String[] args)
    throws InterruptedException {

    int[][] num = new int[6][10];

    int fila;
    int columna;

    int minimo = Integer.MAX_VALUE;
    int filaMinimo = 0;
    int columnaMinimo = 0;

    int maximo = Integer.MIN_VALUE;
    int filaMaximo = 0;
```

```
    int columnaMaximo = 0;

    System.out.print("\n      ");
    for(columna = 0; columna < 10; columna++) {
      System.out.print("    " + columna + "  ");
    }
    System.out.println();

    System.out.print("    ┌");
    for(columna = 0; columna < 10; columna++) {
      System.out.print("──────");
    }
    System.out.println("┐");

    for(fila = 0; fila < 6; fila++) {
      System.out.print("  " + fila + " │");
      for(columna = 0; columna < 10; columna++) {
        num[fila][columna] = (int)(Math.random() * 1001);
        System.out.printf("%5d ", num[fila][columna]);
        Thread.sleep(100);

        // Calcula el mínimo y guarda sus coordenadas
        if (num[fila][columna] < minimo) {
          minimo = num[fila][columna];
          filaMinimo = fila;
          columnaMinimo = columna;
        }

        // Calcula el máximo y guarda sus coordenadas
        if (num[fila][columna] > maximo) {
          maximo = num[fila][columna];
          filaMaximo = fila;
          columnaMaximo = columna;
        }
      }
      System.out.println("│");
    }
    System.out.print("    └");
    for(columna = 0; columna < 10; columna++) {
      System.out.print("──────");
    }

    System.out.println("┘\n\nEl máximo es " + maximo + " y está en la fila " + filaMaximo + ",\
 columna " + columnaMaximo);
    System.out.println("El mínimo es " + minimo + " y está en la fila " + filaMinimo + ", colu\
mna " + columnaMinimo);
```

```
  }
}
```

Ejercicio 6

Fichero: S72Exercise06.java

```
/**
 * 7.2 Arrays bidimensionales
 *
 * 6. Modica el programa anterior de tal forma que no se repita ningún número en el array.
 *
 * @author Luis José Sánchez
 */
public class S72Exercise06 {

  public static void main(String[] args)
    throws InterruptedException {

    int[][] num = new int[6][10];

    int fila;
    int columna;

    // Genera el contenido del array sin que se repita ningún valor
    boolean repetido;
    int i;
    int j;

    for(fila = 0; fila < 6; fila++) {
      for(columna = 0; columna < 10; columna++) {
        do {
          num[fila][columna] = (int)(Math.random() * 1001);

          // Comprueba si el número generado ya está en el array.
          repetido = false;
          for (i = 0; i < 10 * fila + columna; i++) {
            if (num[fila][columna] == num[i / 10][i % 10]) {
              repetido = true;
            }
          }
        } while (repetido);
      }
    }
```

```
int minimo = Integer.MAX_VALUE;
int filaMinimo = 0;
int columnaMinimo = 0;

int maximo = Integer.MIN_VALUE;
int filaMaximo = 0;
int columnaMaximo = 0;

System.out.print("\n      ");
for(columna = 0; columna < 10; columna++) {
  System.out.print("   " + columna + "  ");
}
System.out.println();

System.out.print("    ┌");
for(columna = 0; columna < 10; columna++) {
  System.out.print("──────");
}
System.out.println("┐");

for(fila = 0; fila < 6; fila++) {
  System.out.print("  " + fila + " │");
  for(columna = 0; columna < 10; columna++) {
    System.out.printf("%5d ", num[fila][columna]);
    Thread.sleep(100);

    // Calcula el mínimo y guarda sus coordenadas
    if (num[fila][columna] < minimo) {
      minimo = num[fila][columna];
      filaMinimo = fila;
      columnaMinimo = columna;
    }

    // Calcula el máximo y guarda sus coordenadas
    if (num[fila][columna] > maximo) {
      maximo = num[fila][columna];
      filaMaximo = fila;
      columnaMaximo = columna;
    }
  }
  System.out.println("│");
}
System.out.print("    └");
for(columna = 0; columna < 10; columna++) {
  System.out.print("──────");
}
```

```
    System.out.println("┘\n\nEl máximo es " + maximo + " y está en la fila " + filaMaximo + ",\
 columna " + columnaMaximo);
    System.out.println("El mínimo es " + minimo + " y está en la fila " + filaMinimo + ", colu\
mna " + columnaMinimo);
  }
}
```

Ejercicio 7

Fichero: S72Exercise07.java

```
/**
 * 7.2 Arrays bidimensionales
 *
 * 7. Mejora el juego "Busca el tesoro" de tal forma que si hay una mina a una
 *    casilla de distancia, el programa avise diciendo ¡Cuidado! ¡Hay una mina cerca!
 *
 * @author Luis José Sánchez
 */
public class S72Exercise07 {

  // Se definen constantes para representar el contenido de las celdas
  static final int VACIO = 0;
  static final int MINA = 1;
  static final int TESORO = 2;
  static final int INTENTO = 3;

  public static void main(String[] args) {

    int[][] cuadrante = new int[5][4];

    int x;
    int y;

    // Inicializa el array
    for(x = 0; x < 4; x++) {
      for(y = 0; y < 3; y++) {
        cuadrante[x][y] = VACIO;
      }
    }

    // Coloca la mina
    int minaX = (int)(Math.random()*4);
    int minaY = (int)(Math.random()*3);
```

```
cuadrante[minaX][minaY] = MINA;
// Para depuración:
// System.out.println(minaX + ", " + minaY);

// Coloca el tesoro
int tesoroX;
int tesoroY;

do {
  tesoroX = (int)(Math.random()*4);
  tesoroY = (int)(Math.random()*3);
} while ((minaX == tesoroX) && (minaY == tesoroY));

cuadrante[tesoroX][tesoroY] = TESORO;

// Juego
System.out.println("¡BUSCA EL TESORO!");

int oportunidades = 6;
boolean salir = false;
String c = "";

do {
  // Pinta el cuadrante
  for(y = 3; y >= 0; y--) {
    System.out.print(y + "|");
    for(x = 0; x < 5; x++) {
      if (cuadrante[x][y] == INTENTO) {
        System.out.print("X ");
      } else {
        System.out.print("  ");
      }
    }
    System.out.println();
  }
  System.out.println("  ----------\n  0 1 2 3 4\n");

  // Pide las coordenadas
  System.out.print("Coordenada x: ");
  x = Integer.parseInt(System.console().readLine());
  System.out.print("Coordenada y: ");
  y = Integer.parseInt(System.console().readLine());

  // Mira lo que hay en las coordenadas indicadas por el usuario
  switch(cuadrante[x][y]) {
```

```
        case VACIO:
          cuadrante[x][y] = INTENTO;
          if ((Math.abs(x - minaX) < 2) && (Math.abs(y - minaY) < 2)) {
            System.out.println("Cuidado, hay una mina cerca.");
          }
          break;
        case MINA:
          System.out.println("Lo siento, has perdido.");
          salir = true;
          break;
        case TESORO:
          System.out.println("Enhorabuena, has encontrado el tesoro.");
          salir = true;
      }
    } while (!salir);

    // Pinta el cuadrante
    for(y = 3; y >= 0; y--) {
      System.out.print(y + " ");
      for(x = 0; x < 5; x++) {
        switch(cuadrante[x][y]) {
          case VACIO:
            c = "  ";
            break;
          case MINA:
            c = "* ";
            break;
          case TESORO:
            c = "€ ";
            break;
          case INTENTO:
            c = "x ";
            break;
        }
        System.out.print(c);
      }
      System.out.println();
    }
    System.out.println("  ----------\n  0 1 2 3 4\n");
  }
}
```

Ejercicio 8

Fichero: S72Exercise08.java

```
/**
 *
 * 7.2 Arrays bidimensionales
 *
 * 8. Escribe un programa que, dada una posición en un tablero de ajedrez,
 *    nos diga a qué casillas podría saltar un alfil que se encuentra en esa
 *    posición. Como se indica en la figura, el alfil se mueve siempre en
 *    diagonal. El tablero cuenta con 64 casillas.
 *    Las columnas se indican con las letras de la "a" a la "h" y las filas
 *    se indican del 1 al 8.
 *    Ejemplo:
 *    Introduzca la posición del alfil: d5
 *    El alfil puede moverse a las siguientes posiciones: h1 a2 g2 b3 e3 c4 e4 c6 e6 b7 f7 a8 \
g8
 *
 * @author Luis José Sánchez
 *
 */
public class S72Exercise08 {

  public static void main(String[] args) {

    System.out.print("\nIntroduzca la posición del alfil, por ejemplo d5: ");
    String posicionAlfil = System.console().readLine();
    int columnaAlfil = (int)(posicionAlfil.charAt(0)) - 96;
    int filaAlfil = (int)(posicionAlfil.charAt(1)) - 48;

    System.out.println("El alfil puede moverse a las siguientes posiciones:");

    for(int fila = 8; fila >= 1; fila--) {
      for(int columna = 1; columna <= 8; columna++) {
        if ((Math.abs(filaAlfil - fila) == Math.abs(columnaAlfil - columna))
              && (! ((filaAlfil == fila) && (columnaAlfil == columna)))) {
          System.out.print((char)(columna + 96) + "" + fila + " ");
        }
      }
    }
  }
}
```

Ejercicio 8

Fichero: S72Exercise08ConDibujoTablero.java

```java
/**
 *
 * 7.2 Arrays bidimensionales
 *
 * 8. Escribe un programa que, dada una posición en un tablero de ajedrez,
 *    nos diga a qué casillas podría saltar un alfil que se encuentra en esa
 *    posición. Como se indica en la figura, el alfil se mueve siempre en
 *    diagonal. El tablero cuenta con 64 casillas.
 *    Las columnas se indican con las letras de la "a" a la "h" y las filas
 *    se indican del 1 al 8.
 *    Ejemplo:
 *    Introduzca la posición del alfil: d5
 *    El alfil puede moverse a las siguientes posiciones: h1 a2 g2 b3 e3 c4 e4 c6 e6 b7 f7 a8 \
g8
 *
 * @author Luis José Sánchez
 *
 */
public class S72Exercise08ConDibujoTablero {

  public static void main(String[] args) {

    // se definen constantes para representar el
    // contenido de las celdas
    final String RESET = "\033[0m";
    final String INVERSO = "\033[7m";
    final String BLANCA = "  ";
    final String NEGRA = INVERSO + BLANCA + RESET;
    final String ALFIL = "□ ";  // Δ□□□
    final String MOVIMIENTOBLANCO = "• "; // □
    final String MOVIMIENTONEGRO = INVERSO + MOVIMIENTOBLANCO + RESET;

    String[][] tablero = new String[9][9]; // la fila 0 y la columna 0 no se utilizan
    int fila;
    int columna;
    String casilla;

    // pinta el tablero vacío
    System.out.println("\n  □□□□□□□□□□□□□□□□□□□□");
    casilla = BLANCA;
    for(fila = 8; fila >= 1; fila--) {
      System.out.print( fila + " □□");
      for(columna = 1; columna <= 8; columna++) {
        if ((fila % 2) == 1) { // fila impar
          if ((columna % 2) == 1) { // columna impar
            casilla = BLANCA;
```

```
        } else { // columna par
          casilla = NEGRA;
        }
      } else { // fila par
        if ((columna % 2) == 1) { // columna impar
          casilla = NEGRA;
        } else { // columna par
          casilla = BLANCA;
        }
      }
      System.out.print(casilla);
      tablero[fila][columna] = casilla;
    }
    System.out.println("□□");
  }
  System.out.println("  □□□□□□□□□□□□□□□□□□□□");
  System.out.println("     a b c d e f g h");

  // pide las coordenadas
  System.out.print("\nIntroduzca la posición del alfil, por ejemplo d5: ");
  String posicionAlfil = System.console().readLine();
  int columnaAlfil = (int)(posicionAlfil.charAt(0)) - 96;
  int filaAlfil = (int)(posicionAlfil.charAt(1)) - 48;

  if ((((filaAlfil % 2) + (columnaAlfil % 2)) % 2) == 0) {
    tablero[filaAlfil][columnaAlfil] = ALFIL;
  } else {
    tablero[filaAlfil][columnaAlfil] = INVERSO + ALFIL + RESET;
  }

  // pinta el tablero con el alfil y las posiciones hacia donde puede moverse
  System.out.println("\n  □□□□□□□□□□□□□□□□□□□□");
  for(fila = 8; fila >= 1; fila--) {
    System.out.print( fila + " □□");
    for(columna = 1; columna <= 8; columna++) {
      if ((Math.abs(filaAlfil - fila) == Math.abs(columnaAlfil - columna))
            && (! ((filaAlfil == fila) && (columnaAlfil == columna)))) {
        if (tablero[fila][columna] == BLANCA) {
          tablero[fila][columna] = MOVIMIENTOBLANCO;
        } else {
          tablero[fila][columna] = MOVIMIENTONEGRO;
        }
      }
      System.out.print(tablero[fila][columna]);
    }
    System.out.println("□□");
```

```
    }
    System.out.println("  □□□□□□□□□□□□□□□□□□□□");
    System.out.println("     a b c d e f g h");

    System.out.println("\nEl alfil puede moverse a las siguientes posiciones:");

    for(fila = 8; fila >= 1; fila--) {
      for(columna = 1; columna <= 8; columna++) {
        if ((Math.abs(filaAlfil - fila) == Math.abs(columnaAlfil - columna))
              && (! ((filaAlfil == fila) && (columnaAlfil == columna)))) {
          System.out.print((char)(columna + 96) + "" + fila + " ");
        }
      }
    }
  }
}
```

Ejercicio 9

Fichero: S72Exercise09.java

```
/**
 *
 * Aprende Java con Ejercicios (https://leanpub.com/aprendejava)
 *
 * 7.2 Arrays bidimensionales
 *
 * Ejercicio 9
 *
 * @author Luis José Sánchez
 *
 */

public class S72Exercise09 {

  public static void main(String[] args) {

    int[][] n = new int[12][12];
    int capa;
    int i;
    int j;
    int aux1;
    int aux2;
```

```
// genera el array y lo muestra ////////////////////////

System.out.println("Array original");
System.out.println("--------------");

for(i = 0; i < 12; i++) {
  for(j = 0; j < 12; j++) {
    n[i][j] = (int)(Math.random()*101);
    System.out.printf("%5d", n[i][j]);
    }
  System.out.println();
}


// rotación ////////////////////////////////////////////

for(capa = 0; capa < 6; capa++) {

  // rota por arriba
  aux1 = n[capa][11 - capa];
  for (i = 11 - capa; i > capa; i--) {
    n[capa][i] = n[capa][i - 1];
  }

  // rota por la derecha
  aux2 = n[11 - capa][11 - capa];
  for (i = 11 - capa; i > capa + 1; i--) {
    n[i][11 - capa] = n[i - 1][11 - capa];
  }
  n[capa + 1][11 - capa] = aux1;

  // rota por abajo
  aux1 = n[11 - capa][capa];
  for (i = capa; i < 11 - capa - 1; i++) {
    n[11 - capa][i] = n[11 - capa][i + 1];
  }
  n[11 - capa][11 - capa - 1] = aux2;

  // rota por la izquierda
  for (i = capa; i < 11 - capa - 1; i++) {
    n[i][capa] = n[i + 1][capa];
  }
  n[11 - capa - 1][capa] = aux1;

} // for capa
```

```
    // muestra el array resultante ///////////////////////////

    System.out.println("\n\nArray rotado en el sentido de las agujas del reloj");
    System.out.println("---------------------------------------------------");

    for(i = 0; i < 12; i++) {
      for(j = 0; j < 12; j++) {
        System.out.printf("%5d", n[i][j]);
      }
      System.out.println();
    }

  }
}
```

Ejercicio 10

Fichero: S72Exercise10.java

```
/**
 * 7.2 Arrays bidimensionales
 *
 * 10. Realiza el juego de las tres en raya.
 *
 * @author Luis José Sánchez
 */
public class S72Exercise10 {

  static final String JUGADOR = "●";
  static final String ORDENADOR = "X";

  public static void main(String[] args) {

    String[][] tablero = new String[3][3];
    int i;
    int j;
    int movimientos = 0;
    boolean jugadorGana = false;
    boolean ordenadorGana = false;
    String nombreFila = "cba";
    String coordenadas;

    // inicializa el tablero
    for(i = 0; i < 3; i++) {
```

```
      for(j = 0; j < 3; j++) {
        tablero[i][j] = " ";
      }
    }


    // juego /////////////////////////////////////////////////
    do {

      // pinta el tablero
      System.out.println("  □□□□□□□□□□□□□□□");
      for(i = 0; i < 3; i++) {
        System.out.print(nombreFila.charAt(i)+ " □");
        for(j = 0; j < 3; j++) {
          System.out.print("□ " + tablero[i][j] + " ");
          }
        System.out.println("□□");
        System.out.println("  □□□□□□□□□□□□□□□");
      }
      System.out.println("     1   2   3\n");

      // pide las coordenadas
      System.out.print("Introduzca las coordenadas, por ejemplo b2: ");
      coordenadas = System.console().readLine();
      int fila = nombreFila.indexOf(coordenadas.charAt(0));
      int columna = coordenadas.charAt(1) - 1 - 48;
      tablero[fila][columna] = JUGADOR;
      movimientos++;

      // comprueba si gana el jugador
      jugadorGana = (
        (tablero[0][0] == JUGADOR) && (tablero[0][1] == JUGADOR) && (tablero[0][2] == JUGADOR)\
||
        (tablero[1][0] == JUGADOR) && (tablero[1][1] == JUGADOR) && (tablero[1][2] == JUGADOR)\
||
        (tablero[2][0] == JUGADOR) && (tablero[2][1] == JUGADOR) && (tablero[2][2] == JUGADOR)\
||
        (tablero[0][0] == JUGADOR) && (tablero[1][0] == JUGADOR) && (tablero[2][0] == JUGADOR)\
||
        (tablero[0][1] == JUGADOR) && (tablero[1][1] == JUGADOR) && (tablero[2][1] == JUGADOR)\
||
        (tablero[0][2] == JUGADOR) && (tablero[1][2] == JUGADOR) && (tablero[2][2] == JUGADOR)\
||
        (tablero[0][0] == JUGADOR) && (tablero[1][1] == JUGADOR) && (tablero[2][2] == JUGADOR)\
||
        (tablero[0][2] == JUGADOR) && (tablero[1][1] == JUGADOR) && (tablero[2][0] == JUGADOR)
```

```
      );

      if (!jugadorGana && (movimientos < 9)) {
        // juega el ordenador
        do {
          fila = (int)(Math.random() * 3);
          columna = (int)(Math.random() * 3);
        } while (!tablero[fila][columna].equals(" "));

        tablero[fila][columna] = ORDENADOR;
        movimientos++;
        // comprueba si gana el ordenador
        ordenadorGana = (
          (tablero[0][0] == ORDENADOR) && (tablero[0][1] == ORDENADOR) && (tablero[0][2] == OR\
DENADOR) ||
          (tablero[1][0] == ORDENADOR) && (tablero[1][1] == ORDENADOR) && (tablero[1][2] == OR\
DENADOR) ||
          (tablero[2][0] == ORDENADOR) && (tablero[2][1] == ORDENADOR) && (tablero[2][2] == OR\
DENADOR) ||
          (tablero[0][0] == ORDENADOR) && (tablero[1][0] == ORDENADOR) && (tablero[2][0] == OR\
DENADOR) ||
          (tablero[0][1] == ORDENADOR) && (tablero[1][1] == ORDENADOR) && (tablero[2][1] == OR\
DENADOR) ||
          (tablero[0][2] == ORDENADOR) && (tablero[1][2] == ORDENADOR) && (tablero[2][2] == OR\
DENADOR) ||
          (tablero[0][0] == ORDENADOR) && (tablero[1][1] == ORDENADOR) && (tablero[2][2] == OR\
DENADOR) ||
          (tablero[0][2] == ORDENADOR) && (tablero[1][1] == ORDENADOR) && (tablero[2][0] == OR\
DENADOR)
        );
      }

    } while (!jugadorGana && !ordenadorGana && (movimientos < 9));

    // pinta el tablero
    System.out.println("  □□□□□□□□□□□□□□□");
    for(i = 0; i < 3; i++) {
      System.out.print(nombreFila.charAt(i)+ " □");
      for(j = 0; j < 3; j++) {
        System.out.print("□ " + tablero[i][j] + " ");
        }
      System.out.println("□□");
      System.out.println("  □□□□□□□□□□□□□□□");
    }
    System.out.println("    1   2   3\n");
```

```
    if (jugadorGana) {
      System.out.println("¡Enhorabuena! ¡Has ganado!");
    } else if (ordenadorGana) {
      System.out.println("Gana el ordenador.");
    } else {
      System.out.println("Empate. No gana nadie.");
    }
  }
}
```

Ejercicio 11

Fichero: S72Exercise11.java

```
/**
 *
 * Aprende Java con Ejercicios (https://leanpub.com/aprendejava)
 *
 * 7.2 Arrays bidimensionales
 *
 * Ejercicio 11
 *
 * @author Luis José Sánchez
 *
 */
public class S72Exercise11 {
  public static void main(String[] args) {

    int[][] n = new int[10][10];

    // genera el array y lo muestra ////////////////////////
    for(int i = 0; i < 10; i++) {
      for(int j = 0; j < 10; j++) {
        n[i][j] = (int)(Math.random()*101) + 200;
        System.out.printf("%5d", n[i][j]);
        }
      System.out.println();
    }


    System.out.print("\nDiagonal desde la esquina superior izquierda ");
    System.out.println("a la esquina inferior derecha: ");

    int maximo = 200;
    int minimo = 300;
```

```
    int suma = 0;

    for(int i = 0; i < 10; i++) {
      int numero = n[i][i];
      System.out.print(numero + " ");
      if (numero > maximo) {
        maximo = numero;
      }
      if (numero < minimo) {
        minimo = numero;
      }
      suma += numero;
    }

    System.out.println("\nMáximo: " + maximo);
    System.out.println("Mínimo: " + minimo);
    System.out.println("Media: " + ((double)suma / 10));
  }
}
```

Ejercicio 12

Fichero: S72Exercise12.java

```
public class S72Exercise12 {

  public static void main(String[] args) {
    int[][] n = new int[9][9];

    // Genera el array y lo muestra
    for (int i = 0; i < 9; i++) {
      for (int j = 0; j < 9; j++) {
        n[i][j] = (int) (Math.random() * 401) + 500;
        System.out.printf("%5d", n[i][j]);
      }
      System.out.println();
    }

    System.out.print("\nDiagonal desde la esquina inferior izquierda ");
    System.out.println("a la esquina superior derecha: ");

    int maximo = 500;
    int minimo = 900;
    int suma = 0;
```

```
    for (int i = 0; i < 9; i++) {
      int numero = n[8 - i][i];
      System.out.print(numero + " ");
      if (numero > maximo) {
        maximo = numero;
      }
      if (numero < minimo) {
        minimo = numero;
      }
      suma += numero;
    }

    System.out.println("\nMáximo: " + maximo);
    System.out.println("Mínimo: " + minimo);
    System.out.println("Media: " + ((double) suma / 10));
  }

}
```

Ejercicio 13

Fichero: S72Ejercicio13.java

```
public class S72Ejercicio13 {

  public static void main(String[] args) {

    String[] pais = {"España", "Rusia", "Japón", "Australia"};
    int[][] estaturas = new int[4][10];

    // Genera el array de estaturas
    for (int i = 0; i < 4; i++) {
      for (int j = 0; j < 10; j++) {
        estaturas[i][j] = (int) (Math.random() * (210 - 140 + 1)) + 140;
      }
    }

    // Calcula media, mínimo y máximo y muestra los datos

    System.out.printf("%64s\n", "MED MIN MAX");

    for (int i = 0; i < 4; i++) {
      int maximo = 140;
      int minimo = 210;
      int suma = 0;
```

```
      System.out.printf("%9s:", pais[i]);

      for (int estatura : estaturas[i]) {
        System.out.printf("%4d", estatura);
        maximo = estatura > maximo ? estatura : maximo;
        minimo = estatura < minimo ? estatura : minimo;
        suma += estatura;
      }

      System.out.printf(" |%4d%4d%4d\n", suma / 10, minimo, maximo);
    }
  }

}
```

Funciones

Ejercicio 1

```
/**
 * Devuelve verdadero si el número que se pasa como parámetro es capicúa y
   falso en caso contrario.
 * <p>
 * Un número capicúa es el que se lee igual de izquierda a derecha que de
 * derecha a izquierda.
 *
 * @param x número del que se quiere saber si es capicúa
 * @return  verdadero si el número que se pasa como parámetro es capicúa y
 *          falso en caso contrario
 */
public static boolean esCapicua(long x) {
  return x == voltea(x);
}

/**
 * Devuelve verdadero si el número que se pasa como parámetro es capicúa y
 * falso en caso contrario.
 * <p>
 * Un número capicúa es el que se lee igual de izquierda a derecha que de
 * derecha a izquierda.
 *
 * @param x número del que se quiere saber si es capicúa
 * @return  verdadero si el número que se pasa como parámetro es capicúa y
 *          falso en caso contrario
 */
public static boolean esCapicua(int x) {
  return esCapicua((long)x);
}
```

Ejercicio 2

```
/**
 * Devuelve verdadero si el número que se pasa como parámetro es primo y falso
 * en caso contrario.
 * <p>
 * Un número primo es un número natural mayor que 1 y divisible únicamente
 * entre el mismo y entre 1.
 *
 * @param x número del que se quiere saber si es primo
 * @return  verdadero si el número que se pasa como parámetro es primo y falso
 *          en caso contrario
 */
public static Boolean esPrimo(long n) {
  if (n < 2) {
    return false;
  } else {
    for (long i = n / 2; i >= 2; i--) {
      if (n % i == 0) {
        return false;
      }
    }
  }
  return true;
}

/**
 * Devuelve verdadero si el número que se pasa como parámetro es primo y falso
 * en caso contrario.
 * <p>
 * Un número primo es un número natural mayor que 1 y divisible únicamente
 * entre el mismo y entre 1.
 *
 * @param x número del que se quiere saber si es primo
 * @return  verdadero si el número que se pasa como parámetro es primo y falso
 *          en caso contrario
 */
public static Boolean esPrimo(int n) {
  return esPrimo((long) n);
}
```

Ejercicio 3

```java
/**
 * Devuelve el menor primo que es mayor al número que se pasa como parámetro.
 *
 * @param x un número entero
 * @return  el menor primo que es mayor al número que se pasa como parámetro
 */
public static int siguientePrimo(int x) {
  while (!esPrimo(++x)) {};

  return x;
}
```

Ejercicio 4

```java
/**
 * Dada una base y un exponente, devuelve la potencia.
 *
 * @param base      base de la potencia
 * @param exponente exponente de la potencia
 * @return          número resultante de elevar la base a la potencia indicada
 */
public static double potencia(int base, int exponente) {
  if (exponente == 0) {
    return 1;
  }

  if (exponente < 0) {
    return 1/potencia(base, -exponente);
  }

  int n = 1;

  for (int i = 0; i < Math.abs(exponente); i++) {
    n = n * base;
  }

  return n;
}
```

Ejercicio 5

```
/**
 * Cuenta el número de dígitos de un número entero.
 *
 * @param x número al que se le quieren contar los dígitos
 * @return  número de dígitos que tiene el número que se pasa como parámetro
 */
public static int digitos(long x) {
  if (x < 0) {
    x = -x;
  }

  if (x == 0) {
    return 1;
  } else {
    int n = 0;
    while (x > 0) {
      x = x / 10; // se le quita un dígito a x
      n++; // incrementa la cuenta de dígitos
    }
    return n;
  }
}

/**
 * Cuenta el número de dígitos de un número entero.
 *
 * @param x número al que se le quieren contar los dígitos
 * @return  número de dígitos que tiene el número que se pasa como parámetro
 */
public static int digitos(int x) {
  return digitos((long)x);
}
```

Ejercicio 6

```
/**
 * Le da la vuelta a un número.
 *
 * @param x número al que se le quiere dar la vuelta
 * @return  número volteado (al revés)
 */
public static long voltea(long x) {
  if (x < 0) {
    return -voltea(-x);
  }
```

```
    long volteado = 0;

    while(x > 0) {
      volteado = (volteado * 10) + (x % 10);
      x = x / 10;
    }

    return volteado;
  }

  /**
   * Le da la vuelta a un número.
   *
   * @param x número al que se le quiere dar la vuelta
   * @return  número volteado (al revés)
   */
  public static int voltea(int x) {
    return (int)voltea((long)x);
  }
```

Ejercicio 7

```
  /**
  /* Devuelve el dígito que está en la posición <code>n</code> de un número
   * entero. Se empieza contando por el 0 y de izquierda a derecha.
   *
   * @param x número entero
   * @param n posición dentro del número <code>x</code>
   * @return  dígito que está en la posición n del número <code>x</code>
   *          empezando a contar por el 0 y de izquierda a derecha
   */
  public static int digitoN(long x, int n) {
    x = voltea(x);

    while (n-- > 0) {
      x = x / 10;
    }

    return (int)x % 10;
  }

  /**
  /* Devuelve el dígito que está en la posición n de un número entero. Se
   * empieza contando por el 0 y de izquierda a derecha.
   *
   * @param x número entero
```

```
 * @param n posición dentro del número <code>x</code>
 * @return  dígito que está en la posición n del número <code>x</code>
 *          empezando a contar por el 0 y de izquierda a derecha
 */
public static int digitoN(int x, int n) {
  return digitoN((long)x, n);
}
```

Ejercicio 8

```
/**
 * Da la posición de la primera ocurrencia de un dígito dentro de un número
 * entero. Si no se encuentra, devuelve -1.
 *
 * @param x número entero
 * @param d dígito a buscar dentro del número
 * @return  posición de la primera ocurrencia del dígito dentro del número o
 *          -1 si no se encuentra
 */
public static int posicionDeDigito(long x, int d) {
  int i;

  for (i = 0; (i < digitos(x)) && (digitoN(x, i) != d); i++) {};

  if (i == digitos(x)) {
    return -1;
  } else {
    return i;
  }
}

/**
 * Da la posición de la primera ocurrencia de un dígito dentro de un número
 * entero. Si no se encuentra, devuelve -1.
 *
 * @param x número entero
 * @param d dígito a buscar dentro del número
 * @return  posición de la primera ocurrencia del dígito dentro del número o
 *          -1 si no se encuentra
 */
public static int posicionDeDigito(int x, int d) {
  return posicionDeDigito((long)x, d);
}
```

Ejercicio 9

```
/**
 * Le quita a un número <code>n</code> dígitos por detrás (por la derecha).
 *
 * @param x número entero
 * @param n número de dígitos que se le van a quitar
 * @return  número inicial <code>x</code> con <code>n</code> dígitos menos
 *          quitados de la derecha
 */
public static long quitaPorDetras(long x, int n) {
  return x / (long)potencia(10, n);
}

/**
 * Le quita a un número <code>n</code> dígitos por detrás (por la derecha).
 *
 * @param x número entero
 * @param n número de dígitos que se le van a quitar
 * @return  número inicial <code>x</code> con <code>n</code> dígitos menos
 *          quitados de la derecha
 */
public static int quitaPorDetras(int x, int n) {

  return (int)quitaPorDetras((long) x, n);
}
```

Ejercicio 10

```
/**
 * Le quita a un número <code>n</code> dígitos por delante (por la izquierda).            \
                                 *
 * @param x número entero
 * @param n número de dígitos que se le van a quitar
 * @return  número inicial <code>x</code> con <code>n</code> dígitos menos
 *          quitados de la izquierda
 */
public static long quitaPorDelante(long x, int n) {
  x = pegaPorDetras(x, 1); // "cierra" el número por si acaso termina en 0
  x = voltea(quitaPorDetras(voltea(x), n));
  x = quitaPorDetras(x, 1);
  return x;
}

/**
 * Le quita a un número <code>n</code> dígitos por delante (por la izquierda).            \
                                 *
 * @param x número entero
```

```
 * @param n número de dígitos que se le van a quitar
 * @return  número inicial <code>x</code> con <code>n</code> dígitos menos
 *          quitados de la izquierda
 */
public static int quitaPorDelante(int x, int n) {
  return (int)quitaPorDelante((long)x, n);
}
```

Ejercicio 11

```
/**
 * Añade un dígito a un número por detrás (por la derecha).
 *
 * @param x número entero
 * @param d dígito que se le va a pegar por la derecha
 * @return  número inicial <code>x</code> con el dígito <code>d</code> pegado
 *          por la derecha
 */
public static long pegaPorDetras(long x, int d) {
  return juntaNumeros(x, d);
}

/**
 * Añade un dígito a un número por detrás (por la derecha).
 *
 * @param x número entero
 * @param d dígito que se le va a pegar por la derecha
 * @return  número inicial <code>x</code> con el dígito <code>d</code> pegado
 *          por la derecha
 */
public static int pegaPorDetras(int x, int d) {
  return (int)pegaPorDetras((long)x, d);
}
```

Ejercicio 12

```
/**
 * Añade un dígito a un número por delante (por la izquierda).
 *
 * @param x número entero
 * @param d dígito que se le va a pegar por la izquierda
 * @return  número inicial <code>x</code> con el dígito <code>d</code> pegado
 *          por la izquierda
 */
public static long pegaPorDelante(long x, int d) {
  return juntaNumeros(d, x);
}

/**
 * Añade un dígito a un número por delante (por la izquierda).
 *
 * @param x número entero
 * @param d dígito que se le va a pegar por la izquierda
 * @return  número inicial <code>x</code> con el dígito <code>d</code> pegado
 *          por la izquierda
 */
public static int pegaPorDelante(int x, int d) {
  return (int)pegaPorDelante((long)x, d);
}
```

Ejercicio 13

```
/**
 * Toma como parámetros las posiciones inicial y final dentro de un número y
 * devuelve el trozo correspondiente.
 * <p>
 * Las posiciones se cuentan de izquierda a derecha comenzando por el cero.
 *
 * @param x      número entero
 * @param inicio posición inicial
 * @param fin    posición final
 * @return       trozo de número compuesto por todos los dígitos que van desde
 *               la posición inicial a la posición final incluyendo ambos
 */
public static long trozoDeNumero(long x, int inicio, int fin) {
  int longitud = digitos(x);
  x = quitaPorDelante(x, inicio);
  x = quitaPorDetras(x, longitud - fin - 1);
  return x;
}

/**
```

```
 * Toma como parámetros las posiciones inicial y final dentro de un número y
 * devuelve el trozo correspondiente.
 * <p>
 * Las posiciones se cuentan de izquierda a derecha comenzando por el cero.
 *
 * @param x      número entero
 * @param inicio posición inicial
 * @param fin    posición final
 * @return       trozo de número compuesto por todos los dígitos que van desde
 *               la posición inicial a la posición final incluyendo ambos
 */
public static int trozoDeNumero(int x, int inicio, int fin) {
  return (int)trozoDeNumero((long)x, inicio, fin);
}
```

Ejercicio 14

```
/**
 * Pega dos números para formar uno solo.
 *
 * @param x trozo que se pegará por la izquierda
 * @param y trozo que se pegará por la derecha
 * @return  número compuesto de los trozos <code>x</code> e <code>y</code>
 */
public static long juntaNumeros(long x, long y) {
  return (long)(x * potencia(10, digitos(y))) + y;
}

/**
 * Pega dos números para formar uno solo.
 *
 * @param x trozo que se pegará por la izquierda
 * @param y trozo que se pegará por la derecha
 * @return  número compuesto de los trozos <code>x</code> e <code>y</code>
 */
public static int juntaNumeros(int x, int y) {
  return (int)(juntaNumeros((long)x, (long)y));
}
```

Programa que prueba los ejercicios del 1 al 14

Fichero: S08Ejercicio01a14.java

```
/**
 * 8. Funciones
 *
 * Ejercicios 1 a 14
 *
 * @author Luis José Sánchez
 */

import matematicas.Varias;

public class S08Ejercicio01a14 {

  public static void main(String[] args) {

    // esCapicua ///////////////////////////////////////////////////

    if (matematicas.Varias.esCapicua(29)) {
      System.out.println("El 29 es capicúa");
    }

    if (matematicas.Varias.esCapicua(464)) {
      System.out.println("El 464 es capicúa");
    }

    // esPrimo /////////////////////////////////////////////////////

    if (matematicas.Varias.esPrimo(29)) {
      System.out.println("El 29 es primo");
    }

    if (matematicas.Varias.esPrimo(80)) {
      System.out.println("El 80 es primo");
    }

    // siguientePrimo ///////////////////////////////////////////////

    System.out.println("El siguiente primo mayor a 23 es " + matematicas.Varias.siguientePrimo\
(23));
    System.out.println("El siguiente primo mayor a 100 es " + matematicas.Varias.siguientePrim\
o(100));

    // potencia ////////////////////////////////////////////////////

    System.out.println("2^10 = " + matematicas.Varias.potencia(2, 10));
    System.out.println("5^(-3) = " + matematicas.Varias.potencia(5, -3));
    System.out.println("10^6 = " + matematicas.Varias.potencia(10, 6));
```

```
    // digitos /////////////////////////////////////////////////////

    System.out.println("El número 0 tiene " + matematicas.Varias.digitos(0) + " dígito/s.");
    System.out.println("El número 7 tiene " + matematicas.Varias.digitos(7) + " dígito/s.");
    System.out.println("El número 674893123 tiene " + matematicas.Varias.digitos(674893123) + \
" dígito/s.");

    // voltea //////////////////////////////////////////////////////

    System.out.println("El 5 volteado es " + matematicas.Varias.voltea(5));
    System.out.println("El 398004321 volteado es " + matematicas.Varias.voltea(398004321));
    System.out.println("El -75839 volteado es " + matematicas.Varias.voltea(-75839));

    // digitoN /////////////////////////////////////////////////////

    System.out.println("En la posición 0 del 3452 está el " + matematicas.Varias.digitoN(3452,\
 0));
    System.out.println("En la posición 6 del 857964034 está el " + matematicas.Varias.digitoN(\
857964034, 6));
    System.out.println("En la posición 4 del 857964034 está el " + matematicas.Varias.digitoN(\
857964034, 4));

    // posicionDeDigito ////////////////////////////////////////////

    System.out.println("En el 3452, el dígito 4 está en la posición " + matematicas.Varias.pos\
icionDeDigito(3452, 4));
    System.out.println("En el 78604321, el dígito 1 está en la posición " + matematicas.Varias\
.posicionDeDigito(78604321, 1));
    System.out.println("En el 78604321, el dígito 7 está en la posición " + matematicas.Varias\
.posicionDeDigito(78604321, 7));
    System.out.println("En el 78604321, el dígito 5 está en la posición " + matematicas.Varias\
.posicionDeDigito(78604321, 5));

    // quitaPorDetras //////////////////////////////////////////////

    System.out.println("Si al 78604321 se le quitan por detrás 4 dígitos, se queda como " + ma\
tematicas.Varias.quitaPorDetras(78604321, 4));
    System.out.println("Si al 1000 se le quita por detrás 1 dígito, se queda como " + matemati\
cas.Varias.quitaPorDetras(1000, 1));

    // quitaPorDelante /////////////////////////////////////////////

    System.out.println("Si al 78604321 se le quitan por delante 4 dígitos, se queda como " + m\
atematicas.Varias.quitaPorDelante(78604321, 4));
    System.out.println("Si al 78604000 se le quitan por delante 2 dígitos, se queda como " + m\
```

```
atematicas.Varias.quitaPorDelante(78604000, 2));

    // pegaPorDetras ///////////////////////////////////////////////

    System.out.println("Si al 567 se le pega por detrás el 1 da el " + matematicas.Varias.pega\
PorDetras(567, 1));
    System.out.println("Si al 33 se le pega por detrás el 0 da el " + matematicas.Varias.pegaP\
orDetras(33, 0));

    // pegaPorDelante //////////////////////////////////////////////

    System.out.println("Si al 567 se le pega por delate el 1 da el " + matematicas.Varias.pega\
PorDelante(567, 1));
    System.out.println("Si al 33 se le pega por delante el 0 da el " + matematicas.Varias.pega\
PorDelante(33, 0));

    // trozoDeNumero ///////////////////////////////////////////////

    System.out.println("Al 78604000 le cojo el trozo que va de la posición 0 a la 3: " + matem\
aticas.Varias.trozoDeNumero(78604000, 0, 3));
    System.out.println("Al 78604000 le cojo el trozo que va de la posición 4 a la 6: " + matem\
aticas.Varias.trozoDeNumero(78604000, 4, 6));
    System.out.println("Al 78604000 le cojo el trozo que va de la posición 2 a la 2: " + matem\
aticas.Varias.trozoDeNumero(78604000, 2, 2));

    // juntaNumeros ////////////////////////////////////////////////

    System.out.println("Juntando el 21 y el 40 da el " + matematicas.Varias.juntaNumeros(21, 4\
0));
    System.out.println("Juntando el 789 y el 250 da el " + matematicas.Varias.juntaNumeros(789\
, 250));

  }
}
```

Ejercicio 15

Fichero: S08Ejercicio15.java

```
/**
 * 8. Funciones
 *
 * 15. Muestra los números primos que hay entre 1 y 1000.
 *
 * @author Luis José Sánchez
 */
import matematicas.Varias;

public class S08Ejercicio15 {

  public static void main(String[] args) {

    for(int i = 1; i < 1001; i++) {
      if (matematicas.Varias.esPrimo(i)) {
        System.out.print(i + "  ");
      }
    }
  }
}
```

Ejercicio 16

Fichero: S08Ejercicio16.java

```
/**
 * 8. Funciones
 *
 * 16. Muestra los números capicúa que hay entre 1 y 99999.
 *
 * @author Luis José Sánchez
 */
import matematicas.Varias;

public class S08Ejercicio16 {

  public static void main(String[] args) {

    for(int i = 1; i < 99999; i++) {
      if (matematicas.Varias.esCapicua(i)) {
        System.out.print(i + "  ");
      }
    }
  }
}
```

Ejercicio 17

Fichero: S08Ejercicio17.java

```
/**
 * 8. Funciones
 *
 * 17. Escribe un programa que pase de binario a decimal.
 *
 * @author Luis José Sánchez
 */
import matematicas.Varias;

public class S08Ejercicio17 {

  public static void main(String[] args) {

    long decimal = 0;

    System.out.print("Introduzca un número binario: ");
    long binario = Long.parseLong(System.console().readLine());

    int bits = matematicas.Varias.digitos(binario);

    for(int i = 0; i < bits; i++) {
      decimal += matematicas.Varias.digitoN(binario, bits - i - 1) * matematicas.Varias.potenc\
ia(2, i);
    }

    System.out.println(binario + " en binario es " + decimal + " en decimal.");
  }
}
```

Ejercicio 18

Fichero: S08Ejercicio18.java

```java
/**
 * 8. Funciones
 *
 * 18. Escribe un programa que pase de decimal a binario.
 *
 * @author Luis José Sánchez
 */
import matematicas.Varias;

public class S08Ejercicio18 {

  public static void main(String[] args) {

    System.out.print("Introduzca un número en base diez para pasarlo a binario: ");
    int decimal = Integer.parseInt(System.console().readLine());

    System.out.println(decimal + " en decimal es " + decimalABinario(decimal) + " en binario."\
);

  } // main

  /**
   * Pasa un número decimal (en base 10) a binario (base 2).
   *
   * @param decimal número entero en base 10
   * @return        número inicial pasado a binario
   */
  public static long decimalABinario(int decimal) {

    if (decimal == 0) {
      return 0;
    }

    long binario = 1;

    while (decimal > 1) {
      binario = matematicas.Varias.pegaPorDetras(binario, decimal % 2);
      decimal = decimal / 2;
    }
    binario = matematicas.Varias.pegaPorDetras(binario, 1);
    binario = matematicas.Varias.voltea(binario);
    binario = matematicas.Varias.quitaPorDetras(binario, 1);

    return binario;
  }
}
```

Ejercicio 19

Fichero: S08Ejercicio19.java

```java
/**
 * 8. Funciones
 *
 * 19. Une y amplía los dos programas anteriores de tal forma que se permita
 *     convertir un número entre cualquiera de las siguientes bases: decimal,
 *     binario, hexadecimal y octal.
 *
 * @author Luis José Sánchez
 */
import matematicas.Varias;

public class S08Ejercicio19 {

  public static void main(String[] args) {

    String resultado = "";
    long numeroIntroducido = 666;

    System.out.println(" 1) Binario -> Octal");
    System.out.println(" 2) Binario -> Decimal");
    System.out.println(" 3) Binario -> Hexadecimal");
    System.out.println(" 4) Octal -> Binario");
    System.out.println(" 5) Octal -> Decimal");
    System.out.println(" 6) Octal -> Hexadecimal");
    System.out.println(" 7) Decimal -> Binario");
    System.out.println(" 8) Decimal -> Octal");
    System.out.println(" 9) Decimal -> Hexadecimal");
    System.out.println("10) Hexadecimal -> Binario");
    System.out.println("11) Hexadecimal -> Octal");
    System.out.println("12) Hexadecimal -> Decimal");
    System.out.print("Elija una opción: ");
    int opcion = Integer.parseInt(System.console().readLine());

    System.out.print("Introduzca el número que quiere convertir: ");
    String numeroIntroducidoString = System.console().readLine();

    if (opcion < 10) {
      numeroIntroducido = Long.parseLong(numeroIntroducidoString);
    }

    switch (opcion) {
      case 1:
```

```
        resultado = Long.toString(binarioAOctal(numeroIntroducido));
        break;
      case 2:
        resultado = Long.toString(binarioADecimal(numeroIntroducido));
        break;
      case 3:
        resultado = binarioAHexadecimal(numeroIntroducido);
        break;
      case 4:
        resultado = Long.toString(octalABinario(numeroIntroducido));
        break;
      case  5:
        resultado = Long.toString(binarioADecimal(octalABinario(numeroIntroducido)));
        break;
      case  6:
        resultado = binarioAHexadecimal(octalABinario(numeroIntroducido));
        break;
      case  7:
        resultado = Long.toString(decimalABinario(numeroIntroducido));
        break;
      case  8:
        resultado = Long.toString(binarioAOctal(decimalABinario(numeroIntroducido)));
        break;
      case  9:
        resultado = binarioAHexadecimal(decimalABinario(numeroIntroducido));
        break;
      case 10:
        resultado = Long.toString(hexadecimalABinario(numeroIntroducidoString));
        break;
      case 11:
        resultado = Long.toString(binarioAOctal(hexadecimalABinario(numeroIntroducidoString)));
        break;
      case 12:
        resultado = Long.toString(binarioADecimal(hexadecimalABinario(numeroIntroducidoString)\
));
        break;
    }

    System.out.println(resultado);
  } // main

  /**
   * Pasa un número binario (en base 2) a octal (base 8).
   *
   * @param binario número entero en binario
   * @return        número inicial pasado a octal
```

```
   */
  public static long binarioAOctal(long binario) {
    long octal = 1;

    while (binario > 0) {
      octal = octal * 10 + (binarioADecimal(binario % 1000));
      binario = binario / 1000;
    };

    octal = matematicas.Varias.pegaPorDetras(octal, 1);
    octal = matematicas.Varias.voltea(octal);
    octal = matematicas.Varias.quitaPorDetras(octal, 1);
    octal = matematicas.Varias.quitaPorDelante(octal, 1);

    return octal;
  }

  /**
   * Pasa un número binario (en base 2) a decimal (base 10).
   *
   * @param binario número entero en binario
   * @return        número inicial pasado a decimal
   */
  public static long binarioADecimal(long binario) {
    long decimal = 0;

    int bits = matematicas.Varias.digitos(binario);

    for(int i = 0; i < bits; i++) {
      decimal += matematicas.Varias.digitoN(binario, bits - i - 1) * matematicas.Varias.potenc\
ia(2, i);
    }

    return decimal;
  }

  /**
   * Pasa un número binario (en base 2) a hexadecimal (base 16).
   *
   * @param binario número entero en binario
   * @return        número inicial pasado a hexadecimal
   */
   public static String binarioAHexadecimal(long binario) {
    String hexadecimal = "";
    String digitosHexa = "0123456789ABCDEF";
```

```
    while (binario > 0) {
      hexadecimal = digitosHexa.charAt((int)binarioADecimal(binario % 10000)) + hexadecimal;
      binario = binario / 10000;
    };

    return hexadecimal;
  }

  /**
   * Pasa un número octal (en base 8) a binario (base 2).
   *
   * @param octal número entero en octal
   * @return      número inicial pasado a binario
   */
  public static long octalABinario(long octal) {
    long binario = 0;

    for (int i = 0; i < matematicas.Varias.digitos(octal); i++) {
      binario = binario * 1000 + decimalABinario(matematicas.Varias.digitoN(octal, i));
    }

    return binario;
  }

  /**
   * Pasa un número decimal (en base 10) a binario (base 2).
   *
   * @param octal número entero en decimal
   * @return      número inicial pasado a binario
   */
  public static long decimalABinario(long decimal) {
    if (decimal == 0) {
      return 0;
    }

    long binario = 1;

    while (decimal > 1) {
      binario = matematicas.Varias.pegaPorDetras(binario, (int)decimal % 2);
      decimal = decimal / 2;
    }
    binario = matematicas.Varias.pegaPorDetras(binario, 1);
    binario = matematicas.Varias.voltea(binario);
    binario = matematicas.Varias.quitaPorDetras(binario, 1);

    return binario;
```

```
  }

  /**
   * Pasa un número hexadecimal (en base 10) a binario (base 2).
   *
   * @param hexadecimal número entero en hexadecimal
   * @return            número inicial pasado a binario
   */
  public static long hexadecimalABinario(String hexadecimal) {
    String digitosHexa = "0123456789ABCDEF";
    long binario = 0;

    for (int i = 0; i < hexadecimal.length(); i++) {
      binario = binario * 10000 + decimalABinario(digitosHexa.indexOf(hexadecimal.charAt(i)));
    }

    return binario;
  }
}
```

Ejercicio 20

```
  /**
   * Crea un array y lo rellena con valores aleatorios dentro de un rango.
   * <p>
   * Por ejemplo, <code>generaArrayInt(100, 10, 30)</code> devolvería un array
   * de 100 números generados al azar comprendidos entre 10 y 30.
   *
   * @param n      número de elementos que contendrá el array
   * @param minimo límite inferior del intervalo de números aleatorios
   * @param maximo límite superior del intervalo de números aleatorios
   * @return       array relleno con valores aleatorios dentro del rango
   *               definido por <code>minimo</code> y <code>maximo</code>
   */
  public static int[] generaArrayInt(int n, int minimo, int maximo) {
    int[] x = new int[n];

    for(int i = 0; i < n; i++) {
      x[i] = (int)(Math.random()*(maximo - minimo + 1) + minimo);
    }

    return x;
  }
```

Ejercicio 21

```
/**
 * Devuelve el mínimo (el número más pequeño) del array que se pasa como
 * parámetro.
 *
 * @param x array unidimiensional de números enteros
 * @return  el número más pequeño encontrado en el array que se pasa como
 *          parametro
 */
public static int minimoArrayInt(int[] x) {
  int minimo = Integer.MAX_VALUE;

  for (int n : x) {
    if (n < minimo) {
      minimo = n;
    }
  }
  return minimo;
}
```

Ejercicio 22

```
/**
 * Devuelve el máximo (el número más grande) del array que se pasa como
 * parámetro.
 *
 * @param x array unidimiensional de números enteros
 * @return  el número más grande encontrado en el array que se pasa como
 *          parametro
 */
public static int maximoArrayInt(int[] x) {
  int maximo = Integer.MIN_VALUE;

  for (int n : x) {
    if (n > maximo) {
      maximo = n;
    }
  }

  return maximo;
}
```

Ejercicio 23

```
/**
 * Devuelve la media aritmética de los números contenidos en el array que se
 * pasa como parámetro.
 *
 * @param x array unidimiensional de números enteros
 * @return  media aritmética de los números contenidos en el array que se pasa
 *          como parámetro
 */
public static double mediaArrayInt(int[] x) {
  int suma = 0;

  for (int n : x) {
    suma += n;
  }

  return (double)suma / x.length;
}
```

Ejercicio 24

```
/**
 * Nos dice si un determindado valor está o no dentro de un array de números
 * enteros.
 *
 * @param x array unidimiensional de números enteros
 * @param n números entero que se buscará dentro del array
 * @return  verdadero si <code>n</code> se encuentra en el array
 *          <code>x</code> en caso contrario.
 */
public static boolean estaEnArrayInt(int[] x, int n) {
  for (int numero : x) {
    if (numero == n) {
      return true;
    }
  }
  return false;
}
```

Ejercicio 25

```
/**
 * Devuelve la posición (el índice) de la primera ocurrencia de un número
 * dentro de un array.
 *
 * @param x array unidimiensional de números enteros
 * @param n números entero que se buscará dentro del array
 * @return  posición (índice) de la primera ocurrencia del número
 *          <code>n</code> dentro del array <code>x</code> o -1 en caso de
 *          no encontrarse el número
 */
public static int posicionEnArrayInt(int[] x, int n) {
  for(int i = 0; i < x.length; i++) {
    if (x[i] == n) {
      return i;
    }
  }
  return -1;
}
```

Ejercicio 26

```
/**
 * Le da la vuelta a un array. Los primeros números estarán al final y
 * viceversa.
 *
 * @param x array unidimiensional de números enteros
 * @return  array volteada con los primeros números al final y viceversa
 */
public static int[] volteaArrayInt(int[] x) {
  int[] a = new int[x.length];

  for(int i = 0; i < x.length; i++) {
    a[x.length - i - 1] = x[i];
  }

  return a;
}
```

Ejercicio 27

```
/**
 * Devuelve un array rotada <code>n</code> posiciones a la derecha tomando
 * como referencia el array que se pasa como parámetro. Los números que van
 * saliendo por la derecha vuelven a entrar por la izquierda.
 *
 * @param x array unidimiensional de números enteros
 * @param n número de movimientos (rotaciones hacia la derecha) a realizar
 * @return  array rotada <code>n</code> posiciones a la derecha
 */
public static int[] rotaDerechaArrayInt(int[] x, int n) {

  int[] a = x.clone(); // clona en a el contenido de x
  int i;
  int aux;

  while (n-- > 0) {
    aux = a[a.length - 1];
    for(i = a.length - 1; i > 0; i--) {
      a[i] = a[i - 1];
    }
    a[0] = aux;
  }

  return a;
}
```

Ejercicio 28

```
/**
 * Devuelve un array rotada <code>n</code> posiciones a la izquierda tomando
 * como referencia el array que se pasa como parámetro. Los números que van
 * saliendo por la izquierda vuelven a entrar por la derecha.
 *
 * @param x array unidimiensional de números enteros
 * @param n número de movimientos (rotaciones hacia la izquierda) a realizar
 * @return  array rotada <code>n</code> posiciones a la izquierda
 */
public static int[] rotaIzquierdaArrayInt(int[] x, int n) {

  int[] a = x.clone(); // clona en a el contenido de x
  int i;
  int aux;

  while (n-- > 0) {
    aux = a[0];
    for(i = 0; i < a.length - 1; i++) {
```

```
        a[i] = a[i + 1];
      }
      a[a.length - 1] = aux;
    }

    return a;
  }
```

Programa que prueba los ejercicios del 20 al 28

Fichero: S08Ejercicio20a28.java

```
/**
 *
 * 8. Funciones
 *
 * Ejercicios 20-28
 *
 * @author Luis José Sánchez
 *
 */

import array.Array;

public class S08Ejercicio20a28 {

  public static void main(String[] args) {

    int[] a = array.Array.generaArrayInt(20, 0, 100);

    array.Array.muestraArrayInt(a);
    System.out.println("Mínimo: " + array.Array.minimoArrayInt(a));
    System.out.println("Máximo: " + array.Array.maximoArrayInt(a));
    System.out.println("Media: " + array.Array.mediaArrayInt(a));

    if (array.Array.estaEnArrayInt(a, 24)) {
      System.out.println("El 24 está en el array.");
    } else {
      System.out.println("El 24 no está en el array.");
    }

    System.out.println("El 24 está en la posición " + array.Array.posicionEnArrayInt(a, 24));

    System.out.print("Array al revés: ");
    array.Array.muestraArrayInt(array.Array.volteaArrayInt(a));
```

```
    System.out.print("Array rotado 3 pasos a la derecha: ");
    array.Array.muestraArrayInt(array.Array.rotaDerechaArrayInt(a, 3));

    System.out.print("Array rotado 4 pasos a la izquierda: ");
    array.Array.muestraArrayInt(array.Array.rotaIzquierdaArrayInt(a, 4));
  }
}
```

Ejercicio 29

```
  /**
   * Crea un array bidimensional de números enteros y lo rellena con valores
   * aleatorios dentro de un rango.
   * <p>
   * Por ejemplo, <code>generaArrayBiInt(8, 7, 10, 30)</code> devolvería un
   * array de 8 filas por 7 columnas relleno con números generados al azar
   * comprendidos entre 10 y 30.
   *
   * @param filas    número de filas que tendrá el array
   * @param columnas número de columnas que tendrá el array
   * @param minimo   límite inferior del intervalo de números aleatorios
   * @param máximo   límite superior del intervalo de números aleatorios
   * @return         array bidimensional de números enteros rellena con valores
   *                 aleatorios dentro del rango definido por los valores
   *                 <code>minimo</code> y <code>maximo</code>
   */
  public static int[][] generaArrayBiInt(int filas, int columnas, int minimo, int maximo) {
    int[][] x = new int[filas][columnas];

    for(int i = 0; i < filas; i++) {
      for(int j = 0; j < columnas; j++) {
        x[i][j] = (int)(Math.random()*(maximo - minimo) + minimo + 1);
      }
    }

    return x;
  }
```

Ejercicio 30

```
/**
 * Devuelve una fila (array unidimensional) de un array bidimensional
 * que se pasa como parámetro.
 *
 * @param x    array bidimiensional de números enteros
 * @param fila número de la fila que se quiere extraer del array
 *             <code>x</code>
 * @return     fila en forma de array unidimensional extraida del
 *             array <code>x</code>
 */
public static int[] filaDeArrayBiInt(int x[][], int fila) {
  int[] f = new int[x[0].length];

  for (int c = 0; c < x[0].length; c++) {
    f[c] = x[fila][c];
  }

  return f;
}
```

Ejercicio 31

```
/**
 * Devuelve una columna (array unidimensional) de un array
 * bidimensional que se pasa como parámetro.
 *
 * @param x       array bidimiensional de números enteros
 * @param columna número de la columna que se quiere extraer del array
 *                <code>x</code>
 * @return        columna en forma de array unidimensional extraida
 *                del array <code>x</code>
 */
public static int[] columnaDeArrayBiInt(int x[][], int columna) {
  int[] c = new int[x.length];

  for (int f = 0; f < x.length; f++) {
    c[f] = x[f][columna];
  }

  return c;
}
```

Ejercicio 32

```
/**
 * Devuelve la fila y la columna (en un array con dos elementos) de la
 * primera ocurrencia de un número dentro de un array bidimensional.
 * Si el número no se encuentra en el array, la función devuelve -1.
 *
 * @param x array bidimensional de números enteros
 * @param n número que se buscará dentro del array <code>x</code>
 * @return  array unidimensional de dos elementos que indican la fila
 *          y la columna donde se encuentra <code>n</code> o <code>
 *          {-1, -1}</code> en caso de que <code>n</code> no se
 *          encuentre en <code>x</code>
 */
public static int[] coordenadasEnArrayBiInt(int x[][], int n) {
  for (int f = 0; f < x.length; f++) {
    for (int c = 0; c < x[0].length; c++) {
      if (x[f][c] == n) {
        int[] coordenadas = {f, c};
        return coordenadas;
      }
    }
  }
  int[] coordenadas = {-1, -1};
  return coordenadas;
}
```

Ejercicio 33

```
/**
 * Dice si un número que se encuentra en una posición determinada de
 * una matriz (un array bidimensional) que se pasa como parámetro es o
 * no punto de silla.
 * <p>
 * El punto de silla cumple la condición de ser el mínimo en su fila y
 * máximo en su columna.
 *
 * @param x array bidimensional de números enteros
 * @param i fila dentro del array <code>x</code>
 * @param j columna dentro del array <code>x</code>
 * @return  verdadero si el número que se encuentra en la fila <code>i
 *          </code> y la columna <code>j</code> es el mínimo en su
 *          fila y el máximo en su columna.
 */
public static boolean esPuntoDeSilla(int x[][], int i, int j) {

  int[] fila = filaDeArrayBiInt(x, i);
  int[] columna = columnaDeArrayBiInt(x, j);
```

```
  return ((Array.minimoArrayInt(fila) == x[i][j])
      && (Array.maximoArrayInt(columna) == x[i][j]));
}
```

Ejercicio 34

```
/**
 * Devuelve un array que contiene una de las diagonales del array
 * bidimensional que se pasa como parámetro.
 *
 * @param x         array bidimiensional de números enteros
 * @param fila      fila del número que marcará las dos posibles
 *                  diagonales dentro del array <code>x</code>
 * @param columna   columna del número que marcará las dos posibles
 *                  diagonales dentro del array <code>x</code>
 * @param direccion cadena de caracteres que indica cuál de las dos
 *                  posibles diagonales se devolverá; la cadena <code>
 *                  "nose"</code> indica que se elige la diagonal que
 *                  va del noroeste hacia el sureste, mientras que la
 *                  cadena <code>"neso"</code> indica que se elige la
 *                  diagonal que va del noreste hacia el suroeste
 * @return          array unidimensional que contiene una diagonal
 *                  definida por un número determinado por <code>fila
 *                  </code> y <code>columna</code> y una dirección
 *                  determinada por el parámetro <code>direccion
 *                  </code>
 */
public static int[] diagonal(int x[][], int fila, int columna, String direccion) {

  int elementos = 0, i, j;
  int[] diagonalAux = new int [1000];

  for (i = 0; i < x.length; i++) {
    for (j = 0; j < x[0].length; j++) {
      if ( (((columna - j) == (fila - i)) && (direccion.equals("nose")))
        || (((columna - j) == (i - fila)) && (direccion.equals("neso"))) ) {
        diagonalAux[elementos++] = x[i][j];
      }
    }
  }

  int[] diagonal = new int[elementos];
  for (j = 0; j < elementos; j++) {
    diagonal[j] = diagonalAux[j];
  }
```

```
    return diagonal;
  }
}
```

Programa que prueba los ejercicios del 29 al 34

Fichero: S08Ejercicio29a34.java

```
/**
 *
 * 8. Funciones
 *
 * Ejercicios 29-34
 *
 * @author Luis José Sánchez
 *
 */

import array.ArrayBi;

public class S08Ejercicio29a34 {

  public static void main(String[] args) {

    int[][] a = array.ArrayBi.generaArrayBiInt(5, 8, 0, 100);

    array.ArrayBi.muestraArrayBiInt(a);
    System.out.print("\nFila 2: ");
    array.Array.muestraArrayInt(array.ArrayBi.filaDeArrayBiInt(a, 2));
    System.out.print("\nColumna 6: ");
    array.Array.muestraArrayInt(array.ArrayBi.columnaDeArrayBiInt(a, 6));
    System.out.print("\nCoordenadas del 24 (fila, columna): ");
    array.Array.muestraArrayInt(array.ArrayBi.coordenadasEnArrayBiInt(a, 24));

    int[][] b = {{11, 10, 9}, {4, 5, 7}, {2, 6, 1}};
    array.ArrayBi.muestraArrayBiInt(b);
    System.out.println("\nBusca los puntos de silla: ");
    for(int i = 0; i < 3; i++) {
      for(int j = 0; j < 3; j++) {
        if (array.ArrayBi.esPuntoDeSilla(b, i, j)) {
          System.out.println("fila " + i + ", columna " + j + " -> "  + b[i][j]);
        }
      } // for j
    } // for i
```

```
    array.ArrayBi.muestraArrayBiInt(a);
    System.out.print("\nFila: ");
    int fila = Integer.parseInt(System.console().readLine());
    System.out.print("Columna: ");
    int columna = Integer.parseInt(System.console().readLine());
    System.out.print("Dirección (nose/neso): ");
    String direccion = System.console().readLine();
    System.out.print("\nDiagonal: ");
    array.Array.muestraArrayInt(array.ArrayBi.diagonal(a, fila, columna, direccion));
  }

}
```

Ejercicio 35

Fichero: S08Ejercicio35.java

```
import static matematicas.Varias.digitoN;
import static matematicas.Varias.digitos;

public class S08Ejercicio35 {

  public static void main(String[] args) {
    int[] numeros = {470213, 0, 11, 24, 867024};

    for (int numero : numeros) {
      System.out.print("El " + numero + " es el " + convierteEnPalotes(numero));
      System.out.println(" en el sistema de palotes.");
    }
  }


  public static String convierteEnPalotes(int n) {
    String resultado = "";

    for(int i = 0; i < digitos(n); i++) {
      for (int j = 0; j < digitoN(n, i); j++) {
        resultado += "|";
      }
      if (i < digitos(n) - 1) {
        resultado += "-";
      }
    }

    return resultado;
```

```
  }
}
```

Ejercicio 36

Fichero: S08Ejercicio36.java

```
import static array.Array.generaArrayInt;
import static array.Array.muestraArrayInt;
import static matematicas.Varias.esPrimo;

public class S08Ejercicio36 {

  public static void main(String[] args) {
    int[] numeros = generaArrayInt(20, 1, 100);

    System.out.println("Array original: ");
    muestraArrayInt(numeros);
    System.out.println("Primos: ");
    muestraArrayInt(filtraPrimos(numeros));
  }

  public static int[] filtraPrimos(int x[]) {
    int[] primos = new int[x.length];

    int cuentaPrimos = 0;
    for (int numero : x) {
      if (esPrimo(numero)) {
        primos[cuentaPrimos++] = numero;
      }
    }

    int[] resultado = new int[cuentaPrimos];

    for (int i = 0; i < cuentaPrimos; i++) {
      resultado[i] = primos[i];
    }

    return resultado;
  }
}
```

Ejercicio 37

Fichero: S08Ejercicio37.java

```
import static matematicas.Varias.digitoN;
import static matematicas.Varias.digitos;

public class S08Ejercicio37 {

  public static void main(String[] args) {
    int[] numeros = {213, 0, 11, 24, 86};

    for (int numero : numeros) {
      System.out.print("El " + numero + " es el " + convierteEnMorse(numero));
      System.out.println(" en morse.");
    }
  }

  public static String convierteEnMorse(int n) {
    String[] morse = {
          "_ _ _ _ _", ". _ _ _ _", ". . _ _ _", ". . . _ _", ". . . . _",
      ". . . . .", "_ . . . .", "_ _ . . .", "_ _ _ . .", "_ _ _ _ ."
    };

    String resultado = "";

    for(int i = 0; i < digitos(n); i++) {
      resultado += morse[digitoN(n, i)];
    }

    return resultado;
  }
}
```

Ejercicio 38

Fichero: S08Ejercicio38.java

```
import static array.Array.generaArrayInt;
import static array.Array.muestraArrayInt;
import static matematicas.Varias.esCapicua;

public class S08Ejercicio38 {

  public static void main(String[] args) {
    int[] numeros = generaArrayInt(20, 1, 1000);

    System.out.println("Array original: ");
    muestraArrayInt(numeros);
```

```
    System.out.println("Capicúas: ");
    muestraArrayInt(filtraCapicuas(numeros));
  }

  public static int[] filtraCapicuas(int x[]) {
    int[] capicuas = new int[x.length];

    int cuentaCapicuas = 0;
    for (int numero : x) {
      if (esCapicua(numero)) {
        capicuas[cuentaCapicuas++] = numero;
      }
    }

    int[] resultado = new int[cuentaCapicuas];

    for (int i = 0; i < cuentaCapicuas; i++) {
      resultado[i] = capicuas[i];
    }

    return resultado;
  }

}
```

Ejercicio 39

Fichero: S08Ejercicio39.java

```
import static matematicas.Varias.digitoN;
import static matematicas.Varias.digitos;

public class S08Ejercicio39 {

  public static void main(String[] args) {
    int[] numeros = {470213, 2130, 9, 1124, 86};

    for (int numero : numeros) {
      System.out.println(numero);
      System.out.println(convierteEnPalabras(numero) + "\n");
    }
  }

  public static String convierteEnPalabras(int n) {
    String[] digitos = {
```

```
      "cero", "uno", "dos", "tres", "cuatro",
      "cinco", "seis", "siete", "ocho", "nueve"
    };

    String resultado = "";

    int i;
    for (i = 0; i < digitos(n) - 1; i++) {
      resultado += digitos[digitoN(n, i)] + ", ";
    }
    resultado += digitos[digitoN(n, i)];

    return resultado;
  }

}
```

Ejercicio 40

Fichero: S08Ejercicio40.java

```java
import static array.Array.generaArrayInt;
import static array.Array.muestraArrayInt;
import static matematicas.Varias.posicionDeDigito;

public class S08Ejercicio40 {

  public static void main(String[] args) {
    int[] numeros = generaArrayInt(20, 1, 1000);

    System.out.println("Array original: ");
    muestraArrayInt(numeros);
    System.out.println("Capicúas: ");
    muestraArrayInt(filtraCon7(numeros));
  }

  public static int[] filtraCon7(int x[]) {
    int[] con7 = new int[x.length];

    int numerosCon7 = 0;
    for (int numero : x) {
      if (contieneEl7(numero)) {
        con7[numerosCon7++] = numero;
      }
    }
```

```
    int[] resultado = new int[numerosCon7];

    for (int i = 0; i < numerosCon7; i++) {
      resultado[i] = con7[i];
    }

    return resultado;
  }

  public static boolean contieneEl7(int n) {
    return posicionDeDigito(n, 7) != -1;
  }

  public static boolean contieneEl7(long n) {
    return posicionDeDigito(n, 7) != -1;
  }

}
```

Ejercicio 41

Fichero: S08Ejercicio41.java

```
import java.util.Scanner;

public class S08Ejercicio41 {

  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    System.out.print("Introduzca la altura de la figura: ");
    int altura = Integer.parseInt(s.nextLine());

    for (int i = altura; i > 0; i--) {
      System.out.println(linea('*', i));
    }
  }

  public static String linea(char caracter, int repeticiones) {

    String resultado = "";

    for (int i = 0; i < repeticiones; i++) {
      resultado += caracter;
    }
```

```
    return resultado;
  }
}
```

Ejercicio 42

Fichero: S08Ejercicio42.java

```
import java.util.Scanner;

public class S08Ejercicio42 {

  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    System.out.print("Introduzca la altura de la figura: ");
    int altura = Integer.parseInt(s.nextLine());

    System.out.println(linea('*', altura));

    for (int i = 1; i < altura; i++) {
      System.out.print("*" + linea(' ', altura - i - 2));
      if (i < altura - 1) {
        System.out.println("*");
      }
    }
    System.out.println();
  }

  public static String linea(char caracter, int repeticiones) {

    String resultado = "";

    for (int i = 0; i < repeticiones; i++) {
      resultado += caracter;
    }

    return resultado;
  }
}
```

Ejercicio 43

Fichero: S08Ejercicio43.java

```
import java.util.Scanner;

public class S08Ejercicio43 {

  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    System.out.print("Introduzca la altura de la figura: ");
    int altura = Integer.parseInt(s.nextLine());

    for (int i = 0; i < altura; i++) {
      System.out.print(linea(' ', i));
      System.out.println(linea('*', altura - i));
    }
  }

  public static String linea(char caracter, int repeticiones) {

    String resultado = "";

    for (int i = 0; i < repeticiones; i++) {
      resultado += caracter;
    }

    return resultado;
  }
}
```

Ejercicio 44

Fichero: S08Ejercicio44.java

```
import java.util.Scanner;

public class S08Ejercicio44 {

  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    System.out.print("Introduzca la altura de la figura: ");
    int altura = Integer.parseInt(s.nextLine());

    System.out.println(linea('*', altura));

    for (int i = 1; i < altura; i++) {
      System.out.print(linea(' ', i));
      System.out.print("*" + linea(' ', altura - i - 2));
```

```
      if (i < altura - 1) {
        System.out.println("*");
      }
    }
    System.out.println();
  }

  public static String linea(char caracter, int repeticiones) {

    String resultado = "";

    for (int i = 0; i < repeticiones; i++) {
      resultado += caracter;
    }

    return resultado;
  }
}
```

Ejercicio 45

Fichero: S08Ejercicio45.java

```
import java.util.Scanner;

public class S08Ejercicio45 {

  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    System.out.print("Introduzca la altura de la figura: ");
    int altura = Integer.parseInt(s.nextLine());


    for (int i = 1; i < altura; i++) {
      System.out.print(linea('*', i));
      System.out.print(linea(' ', altura * 2 - i * 2 - 1));
      System.out.println(linea('*', i));
    }

    System.out.println(linea('*', altura * 2 - 1));

  }

  public static String linea(char caracter, int repeticiones) {
```

```
    String resultado = "";

    for (int i = 0; i < repeticiones; i++) {
      resultado += caracter;
    }

    return resultado;
  }

}
```

Ejercicio 46

Fichero: S08Ejercicio46.java

```
import java.util.Scanner;

public class S08Ejercicio46 {

  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    System.out.print("Introduzca la altura de la figura: ");
    int altura = Integer.parseInt(s.nextLine());

    for (int i = 1; i < altura; i++) {
      System.out.print(vertices('*', i));
      System.out.print(linea(' ', altura * 2 - i * 2 - 1));
      System.out.println(vertices('*', i));
    }

    System.out.println(linea('*', altura * 2 - 1));

  }

  public static String linea(char caracter, int longitud) {

    String resultado = "";

    for (int i = 0; i < longitud; i++) {
      resultado += caracter;
    }

    return resultado;
  }
```

```
  public static String vertices(char caracter, int longitud) {

    if (longitud == 1) {
      return "*";
    }

    return "*" + linea(' ', longitud - 2) + "*";
  }

}
```

Ejercicio 47

Fichero: S08Ejercicio47.java

```
public class S08Ejercicio47 {

  public static void main(String[] args) {

    int[] a = {};
    System.out.println("a: " + convierteArrayEnString(a));

    int[] b = {8};
    System.out.println("b: " + convierteArrayEnString(b));

    int[] c = {6, 2, 5, 0, 1};
    System.out.println("b: " + convierteArrayEnString(c));

  }

  public static String convierteArrayEnString(int[] a) {
    String resultado = "";
    for (int digito : a) {
      resultado += digito;
    }
    return resultado;
  }

}
```

Ejercicio 48

Fichero: S08Ejercicio48.java

```
public class S08Ejercicio48 {

  /**
   * @param args the command line arguments
   */
  public static void main(String[] args) {

    int[] a = {8, 9, 0};
    int[] b = {1, 2, 3};

    for (int e : concatena(a, b)) {
      System.out.println(e);
    }

  }

  public static int[] concatena(int[] a, int[] b) {
    int[] resultado = new int[a.length + b.length];

    for (int i = 0; i < a.length; i++) {
      resultado[i] = a[i];
    }

    for (int i = 0; i < b.length; i++) {
      resultado[i + a.length] = b[i];
    }

    return resultado;
  }

}
```

Ejercicio 49

Fichero: S08Ejercicio49.java

```
import java.util.Scanner;

public class S08Ejercicio49 {

  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);

    System.out.print("¿Cuántos términos de la sucesión look and say quiere calcular? ");
    int n = Integer.parseInt(s.nextLine());
```

```
    int[] a = {1};

    for (int i = 1; i <= n; i++) {
      System.out.print(convierteArrayEnString(a));
      if (i < n) {
        System.out.print(", ");
      } else {
        System.out.println();
      }

      int[] resultado = {};

      while (a.length > 0) {
        int[] aux = new int[2];
        aux[0] = repeticionesDelPrimero(a);
        aux[1] = a[0];
        a = cola(a);
        resultado = concatena(resultado, aux);
      }

      a = resultado.clone();
    }
  }

  public static String convierteArrayEnString(int[] a) {
    String resultado = "";
    for (int digito : a) {
      resultado += digito;
    }
    return resultado;
  }

  public static int repeticionesDelPrimero(int[] a) {

    int repeticiones = 0;
    int i = 0;

    while ((i < a.length) && (a[0] == a[i])) {
      repeticiones++;
      i++;
    }

    return repeticiones;
  }
```

```
  public static int[] cola(int[] a) {
    // Si el array está vacío, se devuelve un array vacío
    if (a.length == 0) {
      int[] aux = {};
      return aux;
    }

    int r = repeticionesDelPrimero(a);
    int[] c = new int[a.length - r];
    for (int i = r; i < a.length; i++) {
      c[i - r] = a[i];
    }

    return c;
  }

  public static int[] concatena(int[] a, int[] b) {
    int[] resultado = new int[a.length + b.length];

    for (int i = 0; i < a.length; i++) {
      resultado[i] = a[i];
    }

    for (int i = 0; i < b.length; i++) {
      resultado[i + a.length] = b[i];
    }

    return resultado;
  }
}
```

Ejercicio 50

Fichero: S08Ejercicio50.java

```
public class S08Ejercicio50 {

  public static void main(String[] args) {
    int[] a = {8, 9, 0};
    int[] b = {1, 2, 3};

    for (int n : mezcla(a, b)) {
      System.out.print(n + " ");
    }
```

```
    System.out.println();

    int[] c = {4, 3};
    int[] d = {7, 8, 9, 10};

    for (int n : mezcla(c, d)) {
      System.out.print(n + " ");
    }

    System.out.println();

    int[] e = {8, 9, 0, 3};
    int[] f = {1};

    for (int n : mezcla(e, f)) {
      System.out.print(n + " ");
    }

    System.out.println();
    int[] g = {};
    int[] h = {1, 2, 3};

    for (int n : mezcla(g, h)) {
      System.out.print(n + " ");
    }

    System.out.println();
  }


  public static int[] mezcla(int[] a, int[] b) {
    int[] resultado = new int[a.length + b.length];

    int colocadosDeA = 0;
    int colocadosDeB = 0;
    int i = 0;

    do {
      if (colocadosDeA < a.length) {
        resultado[i++] = a[colocadosDeA++];
      }

      if (colocadosDeB < b.length) {
        resultado[i++] = b[colocadosDeB++];
      }
    } while (i < a.length + b.length);
```

```
    return resultado;
  }
}
```

Ejercicio 51

Fichero: S08Ejercicio51.java

```
import array.Array;
import matematicas.Varias;

public class S08Ejercicio51 {

  public static void main(String[] args) {
    int[] a = Array.generaArrayInt(10, 2, 100);

    System.out.println("Array generado:");
    Array.muestraArrayInt(a);

    System.out.println();

    for (int n : a) {
      if (Varias.esPrimo(n)) {
        System.out.print("El " + n + " es primo");
      } else {
        System.out.print("El " + n + " no es primo");
      }

      if (Varias.esCapicua(n)) {
        System.out.println(" y es capicúa.");
      } else {
        System.out.println(" y no es capicúa.");
      }
    }
  }

}
```

Ejercicio 52

Fichero: S08Ejercicio52.java

```
public class S08Ejercicio52 {

  public static void main(String[] args) {
    int[] a = {111, 222, 333, 444};
    int[] b = {52, 37};
    System.out.println(aleatorioDeArray(a));
    System.out.println(aleatorioDeArray(b));
  }

  public static int aleatorioDeArray(int[] a) {
    return a[(int)(Math.random() * a.length)];
  }

}
```

Ejercicio 53

Fichero: S08Ejercicio53.java

```
public class S08Ejercicio53 {

  public static void main(String[] args) {
    int[][] array = new int[4][6];
    int fila;
    int columna;

    for (fila = 0; fila < 4; fila++) {
      for (columna = 0; columna < 6; columna++) {
        array[fila][columna] = (int) (Math.random() * 201);
        System.out.printf("%5d ", array[fila][columna]);
      }
      System.out.println("");
    }
    System.out.println(nEsimo(array, 3));
    System.out.println(nEsimo(array, 20));
    System.out.println(nEsimo(array, 24));
    System.out.println(nEsimo(array, 23));
  }

  public static int nEsimo(int[][] n, int posicion) {
        int filas = n.length;
    int columnas = n[0].length;

    if ((posicion < 0) || (posicion > filas * columnas - 1)) {
      return -1;
```

```
    } else {
          return n[posicion / columnas][posicion % columnas];
        }
  }

}
```

Ejercicio 54

Fichero: S08Ejercicio54.java

```
public class S08Ejercicio54 {

  public static void main(String[] args) {
    System.out.println(ocurrencias(8, 4672));
    System.out.println(ocurrencias(5, 5251535));
    System.out.println(ocurrencias(2, 123456));

    int[] a = {714, 81, 9, 11};
    System.out.println(ocurrencias(1, a));

    int[] b = {42, 13, 12345, 4};
    System.out.println(ocurrencias(4, b));

    int[] c = {6, 66, 666};
    System.out.println(ocurrencias(6, c));
  }

  public static int ocurrencias(int digito, int n) {
    int repetido = 0;

    while (n > 0) {
      if ((n % 10) == digito) {
        repetido++;
      }
      n = n/10;
    }

    return repetido;
  }

  public static int ocurrencias(int digito, int[] a) {
    int repetido = 0;

    for (int n : a) {
```

```
      repetido += ocurrencias(digito, n);
    }

    return repetido;
  }
}
```

Ejercicio 55

Fichero: S08Ejercicio55.java

```
public class S08Ejercicio55 {

  public static void main(String[] args) {
    String[] a = {"casa", "coche", "sol", "mesa", "mesa", "coche", "ordenador", "sol", "CASA"};

    for (String cadena : sinRepetir(a)) {
      System.out.print(cadena + " ");
    }
  }

  public static String[] sinRepetir(String[] s) {
    String[] vacio = {};
    if (s.length == 0) {
      return vacio;
    }

    String[] resultado = new String[s.length];

    resultado[0] = s[0];
    int j = 1;
    for (int i = 1; i < s.length; i++) {
      if (!contiene(resultado, s[i])) {
        resultado[j++] = s[i];
      }
    }

    return compacta(resultado);
  }

  public static boolean contiene(String[] lista, String cadena) {

    for (String elemento : lista) {
      if ((elemento != null) && (elemento.equals(cadena))) {
        return true;
```

```
      }
    }
    return false;
  }

  /**
   * Quita los elementos nulos consecutivos por el final.
   *
   * @param s array que puede tener nulos al final
   * @return array sin nulos
   */
  public static String[] compacta(String[] s) {

    int i = s.length - 1;

    while (s[i] == null) {
      i--;
    }
    String[] resultado = new String[i + 1];
    System.arraycopy(s, 0, resultado, 0, i + 1);

    return resultado;
  }
}
```

Ejercicio 56

Fichero: S08Ejercicio56.java

```
public class S08Ejercicio56 {

  public static void main(String[] args) {
    int[][] a = {
      {45, 92, 14, 20, 25, 78},
      {35, 72, 24, 45, 42, 60},
      {32, 42, 64, 23, 41, 39},
      {98, 45, 94, 11, 18, 48}
    };

    for (int numero : corteza(a)) {
      System.out.print(numero + " ");
    }
  }

  private static int[] corteza(int[][] n) {
```

```
    int filas = n.length;
    int columnas = n[0].length;

    int[] resultado = new int[2 * filas + 2 * columnas - 4];

    int j = 0;

    // Fila superior
    for (int i = 0; i < columnas - 1; i++) {
      resultado[j++] = n[0][i];
    }

    // Columna derecha
    for (int i = 0; i < filas - 1; i++) {
      resultado[j++] = n[i][columnas - 1];
    }

    // Fila inferior
    for (int i = columnas - 1; i > 0; i--) {
      resultado[j++] = n[filas - 1][i];
    }

    // Columna izquierda
    for (int i = filas - 1; i > 0; i--) {
      resultado[j++] = n[i][0];
    }
    return resultado;
  }

}
```

Programación orientada a objetos

Conceptos de POO

Ejercicio 1

¿Cuáles serían los atributos de la clase PilotoDeFormula1? ¿Se te ocurren algunas instancias de esta clase?

- Atributos de la clase PilotoDeFormula1: nombre, edad, campeonatosGanados, numeroDeCarrerasOficiales, mediaDePuntosPorTemporada
- Instancias de la clase PilotoDeFormula1: fernandoAlonso, felipeMassa, kimiRaikkonen

Ejercicio 2

A continuación tienes una lista en la que están mezcladas varias clases con instancias de esas clases. Para ponerlo un poco más difícil, todos los elementos están escritos en minúscula. Di cuáles son las clases, cuáles las instancias, a qué clase pertenece cada una de estas instancias y cuál es la jerarquía entre las clases: paula, goofy, gardfiel, perro, mineral, caballo, tom, silvestre, pirita, rocinante, milu, snoopy, gato, pluto, animal, javier, bucefalo, pegaso, ayudante_de_santa_claus, cuarzo, laika, persona, pato_lucas.

Clases: Perro, Mineral, Caballo, Pirita, Gato, Animal, Cuarzo, Persona

Jerarquía de clases e instancias:

```
Animal
  |
  |---Perro
  |     |---goofy
  |     |---milu
  |     |---snoopy
  |     |---pluto
  |     |---ayudante_de_santa_claus
  |     |---laika
  |
  |---Caballo
  |     |---rocinante
  |     |---bucefalo
  |     |---pegaso
  |
  |---Gato
  |     |---garfield
```

```
  |     |---tom
  |     |---silvestre
  |
  |---Persona
  |     |---paula
  |     |---javier
  |
  |---pato_lucas

Mineral
  |---Pirita
  |---Cuarzo
```

Ejercicio 3

¿Cuáles serían los atributos de la clase Vivienda? ¿Qué subclases se te ocurren?

- Atributos de Vivienda: metrosCuadrados, numeroDeHabitaciones, tieneGaraje, orientacion
- Subclases de Vivienda: Piso, Adosado, Cortijo

Ejercicio 4

Piensa en la liga de baloncesto, ¿qué 5 clases se te ocurren para representar 5 elementos distintos que intervengan en la liga?

Jugador, Partido, Estadio, Equipo, Arbitro

Ejercicio 5

Haz una lista con los atributos que podría tener la clase caballo. A continuación haz una lista con los posibles métodos (acciones asociadas a los caballos).

Clase Caballo:

- atributos: nombre, raza, color, edad, carrerasGanadas
- métodos: cabalga, rumia, relincha, trota

Ejercicio 6

Lista los atributos de la clase Alumno ¿Sería nombre uno de los atributos de la clase? Razona tu respuesta.

numExpediente, nombre, fechaNacimiento, curso, direccion, telefono

“nombre” es un atributo que guarda el nombre completo con tildes, espacios, mayúsculas y minúsculas, etc. En el programa principal, los objetos de la clase Alumno tendrán nombres como alumno1, alumnoAux, a, pepito… y no pueden tener espacios, tildes, etc.

Ejercicio 7

¿Cuáles serían los atributos de la clase Ventana (de ordenador)? ¿cuáles serían los métodos? Piensa en las propiedades y en el comportamiento de una ventana de cualquier programa.

- Atributos de la clase Ventana: esVisible, tieneElFoco, posicion, anchura, altura, colorDeFondo, titulo, esRedimensionable
- Métodos de la clase Ventana: maximiza, minimiza, restaura, cierra, redimiensiona, mueve

POO en Java

Ejercicio 1

Fichero: PruebaCaballo.java

```
/**
 * Programa que prueba la clase Caballo.
 * @author Luis José Sánchez
 */
public class PruebaCaballo {

  public static void main(String[] args) {

    Caballo a = new Caballo("Avra", "marrón moteado", 6, 24);
    Caballo l = new Caballo("Lykos", "negro", 8, 61);

    System.out.println("Hola, me llamo " + a.getNombre());
    a.cabalga();
    a.relincha();

    System.out.println("Hola, yo soy " + l.getNombre());
    l.rumia();
    l.cabalga();
  }
}
```

Fichero: Caballo.java

```
public class Caballo {

  private String nombre;
  private String color;
  private int edad;
  private int carrerasGanadas;

  Caballo (String n, String c, int e, int cg) {
    this.nombre = n;
    this.color = c;
    this.edad = e;
    this.carrerasGanadas = cg;
  }

  public String getNombre() {
    return this.nombre;
  }

  public void cabalga() {
    System.out.println("Tocotoc tocotoc tocotoc");
  }

  public void relincha() {
    System.out.println("Hiiiiiiieeeeee");
  }

  public void rumia() {
    System.out.println("Ñam ñam ñam");
  }
}
```

Ejercicio 2

Fichero: Bicicleta.java

```
public class Bicicleta extends Vehiculo {

  private int pinones; // número de piñones

  public Bicicleta(int p) {
    super();
    this.pinones = p;
  }

  public void hazCaballito() {
```

```
    System.out.println("Estoy haciendo el caballito");
  }
}
```

Fichero: Vehiculo.java

```
public abstract class Vehiculo {

  // atributos de clase
  private static int kilometrosTotales = 0;
  private static int vehiculosCreados = 0;

  // atributos de instancia
  private int kilometrosRecorridos;

  public Vehiculo() {
    this.kilometrosRecorridos = 0;
  }

  public int getKilometrosRecorridos() {
    return this.kilometrosRecorridos;
  }

  public static int getKilometrosTotales() {
    return Vehiculo.kilometrosTotales;
  }

  /**
   * Hace que un vehículo recorra una distancia determinada.
   * <p>
   * Cuando un vehículo recorre una distancia <code>k</code>, se
   * incrementan su propio cuentakilómetros, es decir, su atributo
   * <code>kilometrosRecorridos</code> y también se incrementa la cuenta
   * global de kilómetros que recorren todos los vehículos, es decir, el
   * atributo de clase <code>kilometrosTotales</code>.
   *
   * @param k kilómetros a recorrer
   */
  public void recorre(int k) {
    this.kilometrosRecorridos += k;
    Vehiculo.kilometrosTotales += k;
  }
}
```

Fichero: PruebaVehiculos.java

```java
/**
 * 2. Crea la clase Vehiculo , así como las clases Bicicleta y Coche
 *    como subclases de la primera. Para la clase Vehiculo , crea los
 *    atributos de clase vehiculosCreados y kilometrosTotales , así como
 *    el atributo de instancia kilometrosRecorridos . Crea también algún
 *    método específico para cada una de las subclases. Prueba las
 *    clases creadas mediante un programa con un menú como el que se
 *    muestra a continuación:
 *    VEHÍCULOS
 *    =========
 *    1. Anda con la bicicleta
 *    2. Haz el caballito con la bicicleta
 *    3. Anda con el coche
 *    4. Quema rueda con el coche
 *    5. Ver kilometraje de la bicicleta
 *    6. Ver kilometraje del coche
 *    7. Ver kilometraje total
 *    8. Salir
 *    Elige una opción (1-8):
 *
 * @author Luis José Sánchez
 */
public class PruebaVehiculos {

  public static void main(String[] args) {

    int opcion = 0;
    int km;

    Bicicleta bhSpeedrom = new Bicicleta(9);
    Coche saab93 = new Coche(1900);

    while (opcion != 8) {
      System.out.println("1. Anda con la bicicleta");
      System.out.println("2. Haz el caballito con la bicicleta");
      System.out.println("3. Anda con el coche");
      System.out.println("4. Quema rueda con el coche");
      System.out.println("5. Ver kilometraje de la bicicleta");
      System.out.println("6. Ver kilometraje del coche");
      System.out.println("7. Ver kilometraje total");
      System.out.println("8. Salir");
      System.out.println("Elige una opción (1-8): ");

      opcion = Integer.parseInt(System.console().readLine());

      switch (opcion) {
```

```
        case 1:
          System.out.print("¿Cuántos kilómetros quiere recorrer? ");
          km = Integer.parseInt(System.console().readLine());
          bhSpeedrom.recorre(km);
          break;
        case 2:
          bhSpeedrom.hazCaballito();
          break;
        case 3:
          System.out.print("¿Cuántos kilómetros quiere recorrer? ");
          km = Integer.parseInt(System.console().readLine());
          saab93.recorre(km);
          break;
        case 4:
          saab93.quemaRueda();
          break;
        case 5:
          System.out.println("La bicicleta lleva recorridos ");
          System.out.println(bhSpeedrom.getKilometrosRecorridos() + " Km");
          break;
        case 6:
          System.out.println("El coche lleva recorridos ");
          System.out.println(saab93.getKilometrosRecorridos() + " Km");
          break;
        case 7:
          System.out.println("Los vehículos llevan recorridos ");
          System.out.println(Vehiculo.getKilometrosTotales() + " Km");
        default:
      } // switch
    } // while
  }
}
```

Fichero: Coche.java

```
public class Coche extends Vehiculo {

  private int cilindrada; // cilindrada en cm3

  public Coche(int c) {
    super();
    this.cilindrada = c;
  }

  public void quemaRueda() {
    System.out.println("Fffshhhhhhhhhhh");
```

```
  }
}
```

Ejercicio 3

Fichero: Ave.java

```
public class Ave extends Animal {

  public Ave() {
    super(Sexo.HEMBRA);
  }

  public Ave(Sexo s) {
    super(s);
  }

  public void ponHuevo() {
    if (this.getSexo() == Sexo.MACHO) {
      System.out.println("Soy macho, no puedo poner huevos");
    } else {
      System.out.println("Ahi va eso... un huevo");
    }
  }

  public void limpiate() {
    System.out.println("Me estoy limpiando las plumas");
  }

  public void vuela() {
    System.out.println("Estoy volando");
  }
}
```

Fichero: PruebaAnimales.java

```
/**
 * 3. Crea las clases Animal, Mamifero, Ave, Gato, Perro, Canario,
 *    Pinguino y Lagarto. Crea, al menos, tres métodos específicos de
 *    cada clase y redefne el/los método/s cuando sea necesario.
 *    Prueba las clases creadas en un programa en el que se instancien
 *    objetos y se les apliquen métodos.
 *
 * @author Luis José Sánchez
 */
public class PruebaAnimales {

  public static void main(String[] args) {
    Pinguino tux = new Pinguino(Sexo.MACHO);
    tux.come("palomitas");
    tux.programa();

    Perro laika = new Perro(Sexo.HEMBRA);
    laika.duerme();
    laika.dameLaPata();
    laika.amamanta();
    laika.cuidaCrias();

    Lagarto godzilla = new Lagarto(Sexo.MACHO);
    godzilla.tomaElSol();
    godzilla.duerme();
  }
}
```

Fichero: Canario.java

```
public class Canario extends Ave {

  public Canario() {}

  public Canario(Sexo s) {
    super(s);
  }

  public void canta() {
    System.out.println("Tralaralariiiiiiii");
  }

  public void caza() {
    System.out.println("Los canarios no cazan");
  }
```

```
  public void pia() {
    System.out.println("Pio pio pio");
  }
}
```

Fichero: Lagarto.java

```
public class Lagarto extends Animal {

  public Lagarto() {}

  public Lagarto(Sexo s) {
    super(s);
  }

  public void tomaElSol() {
    System.out.println("Estoy tomando el Sol");
  }

  public void baniate() {
    System.out.println("Me estoy dando un chapuzón");
  }

  public void escondete() {
    System.out.println("Me he escondido, ya no me puedes ver");
  }
}
```

Fichero: Sexo.java

```
public enum Sexo {
  MACHO, HEMBRA
}
```

Fichero: Animal.java

```
public abstract class Animal {

  private Sexo sexo;

  public Animal () {
    this.sexo = Sexo.MACHO;
  }

  public Animal (Sexo s) {
    this.sexo = s;
  }

  public Sexo getSexo() {
    return this.sexo;
  }

  public void duerme() {
    System.out.println("Zzzzzzz");
  }

  public void come(String comida) {
    System.out.println("Estoy comiendo " + comida);
  }
}
```

Fichero: Perro.java

```
public class Perro extends Mamifero {

  public Perro (Sexo s) {
    super(s);
  }

  public Perro () {
    super(Sexo.HEMBRA);
  }

  public void ladra() {
    System.out.println("Guau guau");
  }

  public void dameLaPata() {
    System.out.println("Toma mi patita");
  }

  public void caza() {
```

```
    System.out.println("Estoy cazando perdices");
  }
}
```

Fichero: Pinguino.java

```
public class Pinguino extends Ave {

  public Pinguino() {}

  public Pinguino(Sexo s) {
    super(s);
  }

  public void vuela() {
    System.out.println("Soy un pingüino, no puedo volar");
  }

  public void programa() {
    System.out.println("Soy un pingüino programador, estoy programando en Java");
  }

  public void nada() {
    System.out.println("Estoy nadando");
  }
}
```

Fichero: Gato.java

```
public class Gato extends Mamifero {

  private String raza;

  public Gato (Sexo s, String r) {
    super(s);
    this.raza = r;
  }

  public Gato (Sexo s) {
    super(s);
    this.raza = "siamés";
  }

  public Gato (String r) {
    super(Sexo.HEMBRA);
```

```
    this.raza = r;
  }

  public Gato () {
    super(Sexo.HEMBRA);
    raza = "siamés";
  }

  public void maulla() {
    System.out.println("Miauuuu");
  }

  public void ronronea() {
    System.out.println("mrrrrrr");
  }

  public void come(String comida) {
    if (comida.equals("pescado")) {
      System.out.println("Hmmmm, gracias");
    } else {
      System.out.println("Lo siento, yo solo como pescado");
    }
  }

  public void peleaCon(Gato contrincante) {
    if (this.getSexo() == Sexo.HEMBRA) {
      System.out.println("no me gusta pelear");
    } else {
      if (contrincante.getSexo() == Sexo.HEMBRA) {
        System.out.println("no peleo contra gatitas");
      } else {
        System.out.println("ven aquí que te vas a enterar");
      }
    }
  }

  public void limpiate() {
    System.out.println("Me estoy lamiendo");
  }

  public void caza() {
    System.out.println("Estoy cazando ratones");
  }
}
```

Fichero: Mamifero.java

```
public abstract class Mamifero extends Animal {

  public Mamifero () {
    super();
  }

  public Mamifero (Sexo s) {
    super(s);
  }

  public void amamanta() {
    if (this.getSexo() == Sexo.MACHO) {
      System.out.println("Soy macho, no puedo amamantar :(");
    } else {
      System.out.println("Toma pecho, hazte grande");
    }
  }

  public void cuidaCrias() {
    System.out.println("Estoy cuidando mis crias");
  }

  public void anda() {
    System.out.println("Estoy andando");
  }
}
```

Ejercicio 4

Fichero: PruebaFraccion.java

```
/**
 * 4. Crea la clase Fracción. Los atributos serán numerador y
 *    denominador. Y algunos de los métodos pueden ser invierte,
 *    simplifica, multiplica, divide, etc.
 *
 * @author Luis José Sánchez
 */
public class PruebaFraccion {
  public static void main(String[] args) {

    Fraccion f1 = new Fraccion(-7,8);
    System.out.println(f1 + " x 5 = "+ f1.multiplica(5));
    System.out.println(f1 + " ^-1 = " + f1.invierte());
```

```
    Fraccion f2 = new Fraccion(3, 5);
    System.out.println(f1 + " x " + f2 + " = " + f1.multiplica(f2));
    System.out.println(f1 + " : " + f2 + " = " + f1.divide(f2));

    Fraccion f3 = new Fraccion(910, -350);
    System.out.println(f3 + " = " + f3.simplifica());
  }
}
```

Fichero: Fraccion.java

```
public class Fraccion {

  private int signo;
  private int numerador;
  private int denominador;

  public Fraccion(int n, int d) {
    if (d == 0) {
      System.out.println("Una fracción no puede tener como denominador el número 0");
    } else {
      if (n * d < 0) {
        this.signo = -1;
      } else {
        this.signo = 1;
      }
      this.numerador = Math.abs(n);
      this.denominador = Math.abs(d);
    }
  }

  int getNumerador(){
    return this.numerador;
  }

  int getDenominador(){
    return this.denominador;
  }

  public String toString() {
    if (signo == -1) {
      return "-" + this.numerador + "/" + this.denominador;
    } else {
      return this.numerador + "/" + this.denominador;
    }
  }
```

```
  /**
   * Devuelve una fracción invertida. Lo que antes era el numerador
   * ahora será el denominador y viceversa.
   *
   * @return fracción invertida
   */
  public Fraccion invierte() {
    return new Fraccion(this.signo * this.denominador, this.numerador);
  }

  /**
   * Devuelve una fracción multiplicada por un escalar (un número)
   * <code>n</code>.
   * <p>
   * Cuando una fracción se multiplica por un número <code>n</code>, el
   * resultado es otra fracción con el mismo denominador que la
   * original.
   * El numerador se obtiene multiplicando <code>n</code> por el
   * numerador de la fracción original.
   *
   * @param n escalar por el que se multiplica la fracción original
   * @return  fracción multiplicada por <code>n</code>
   */
  public Fraccion multiplica(int n) {
    return new Fraccion(this.signo * this.numerador * n, this.denominador);
  }

  /**
   * Devuelve una fracción que es el resultado de multiplicar la
   * fracción original por otra fracción que se pasa como parámetro.
   * <p>
   * Cuando se multiplican dos fracciones, el numerador de la fracción
   * resultante es el resultado de multiplicar los numeradores de las
   * dos fracciones. El denominador de la fracción resultante se calcula
   * de forma análoga.
   *
   * @param f fracción por la que se multiplica la fracción original
   * @return  resultado de multiplicar la fracción original por la
   *          fracción que se pasa como parámetro
   */
  public Fraccion multiplica(Fraccion f) {
    return new Fraccion(this.signo * this.numerador * f.getNumerador(), this.denominador * f.g\
etDenominador());
  }
```

```
  /**
   * Devuelve una fracción dividida entre un escalar (un número) <code>n
   * </code>.
   * <p>
   * Cuando una fracción se divide entre un número <code>n</code>, el
   * resultado es otra fracción con el mismo numerador que la original.
   * El denominador se obtiene multiplicando <code>n</code> por el
   * denominador de la fracción original.
   *
   * @param n escalar entre el que se divide la fracción original
   * @return  fracción dividida entre <code>n</code>
   */
  public Fraccion divide(int n) {
    return new Fraccion(this.signo * this.numerador, this.denominador * n);
  }

  /**
   * Devuelve una fracción que es el resultado de dividir la fracción
   * original entre otra fracción que se pasa como parámetro.
   * <p>
   * Para obtener la división de dos fracciones, el numerador de una
   * fracción se multiplica por el denominador de otra y viceversa.
   *
   * @param f fracción entre la que se quiere dividir la fracción
   *          original
   * @return  resultado de dividir la fracción original entre la
   *          fracción que se pasa como parámetro
   */
  public Fraccion divide(Fraccion f) {
    return new Fraccion(this.signo * this.numerador * f.getDenominador(), denominador * f.getN\
umerador());
  }

  /**
   * Devuelve una fracción que es el resultado de simplificar la
   * fracción original.
   * <p>
   * Para simplificar una fracción, se comprueba si numerador y
   * denominador son divisibles entre el mismo número. En tal caso, los
   * dos se dividen. Se repite el proceso hasta que la fracción que se
   * obtiene es irreducible (no se puede simplificar más).
   *
   * @return  resultado de simplificar (si se puede) la fracción
   *          original, o la misma fracción en caso de que la original
   *          sea irreducible
   */
```

```
  public Fraccion simplifica() {

    int s = this.signo;
    int n = this.numerador;
    int d = this.denominador;

    for (int i = 2; i < Math.min(this.numerador, this.denominador); i++) {
      while (((n % i) == 0) && ((d % i) == 0)) {
        n /= i;
        d /= i;
      }
    }

    return new Fraccion(s * n, d);
  }
}
```

Ejercicio 5

Fichero: PedidosPizza.java

```
public class PedidosPizza {
  public static void main(String[] args) {
    Pizza p1 = new Pizza("margarita", "mediana");
    Pizza p2 = new Pizza("funghi", "familiar");
    p2.sirve();
    Pizza p3 = new Pizza("cuatro quesos", "mediana");
    System.out.println(p1);
    System.out.println(p2);
    System.out.println(p3);
    p2.sirve();
    System.out.println("pedidas: " + Pizza.getTotalPedidas());
    System.out.println("servidas: " + Pizza.getTotalServidas());
  }
}
```

Fichero: Pizza.java

```java
/**
 * 5. Crea la clase Pizza con los atributos y métodos necesarios. Sobre
 *    cada pizza se necesita saber el tamaño - mediana o familiar - el
 *    tipo - margarita, cuatro quesos o funghi - y su estado - pedida o
 *    servida. La clase debe almacenar información sobre el número total
 *    de pizzas que se han pedido y que se han servido. Siempre que se
 *    crea una pizza nueva, su estado es "pedida".
 *
 * @author Luis José Sánchez
 */
public class Pizza {

  private static int totalPedidas = 0;
  private static int totalServidas = 0;

  private String tamano;
  private String tipo;
  private String estado;

  public Pizza(String tipo, String tamano) {
    this.tipo = tipo;
    this.tamano = tamano;
    this.estado = "pedida";
    Pizza.totalPedidas++;
  }

  public String toString() {
    return "pizza " + this.tipo + " " + this.tamano + ", " + this.estado;
  }

  public static int getTotalPedidas() {
    return Pizza.totalPedidas;
  }

  public static int getTotalServidas() {
    return Pizza.totalServidas;
  }

  /**
   * Cambia el estado de la pizza de <code>pedida</code> a <code>servida
   * </code>.
   * <p>
   * En caso de que la pizza se hubiera servido ya y se intenta servir
   * de nuevo, se muestra el mensaje <code>esa pizza ya se ha servido
   * </code>.
   */
```

```
  public void sirve() {
    if (this.estado.equals("pedida")) {
      this.estado = "servida";
      Pizza.totalServidas++;
    } else {
      System.out.println("esa pizza ya se ha servido");
    }
  }
}
```

Ejercicio 6

Fichero: Tiempo.java

```
/**
 * Definición de la clase <code>Tiempo</code>.
 * <p>
 * Un objeto de la clase <code>Tiempo</code> se puede crear de dos
 * maneras diferentes. Se pueden indicar las horas, los minutos y los
 * segundos de la forma <code>new Tiempo(3, 20, 45)</code>, o bien se
 * puede indicar únicamente la cantidad de segundos de la forma <code>
 * new Tiempo(12045)</code> (ambos objetos definen el mismo intervalo
 * de tiempo).
 * <p>
 * Internamente, la clase solo almacena los segundos totales. A partir
 * de éstos, se puede deducir todo lo demás.
 *
 * @author Luis José Sánchez
 */
public class Tiempo {

  private int segundos;

  public Tiempo(int horas, int minutos, int segundos) {
    this.segundos = (horas * 3600) + (minutos * 60) + segundos;
  }

  public Tiempo(int s) {
    this.segundos = s;
  }

  public String toString() {
    int segundos = this.segundos;
    int horas = segundos / 3600;
    segundos -= horas * 3600;
```

```
    int minutos = segundos / 60;
    segundos -= minutos * 60;

    if (this.segundos < 0) {
      return "-(" + (-horas) + "h " + (-minutos) + "m " + (-segundos) + "s)";
    } else {
      return horas + "h " + minutos + "m " + segundos + "s";
    }
  }

  private int getSegundos() {
    return this.segundos;
  }

  public Tiempo suma(Tiempo t) {
    return new Tiempo(this.segundos + t.getSegundos());
  }

  public Tiempo resta(Tiempo t) {
    return new Tiempo(this.segundos - t.getSegundos());
  }
}
```

Fichero: PruebaTiempo.java

```
/**
 * 6. Crea la clase Tiempo con los métodos suma y resta. Los objetos de
 *    la clase Tiempo son intervalos de tiempo y se crean de la forma
 *    Tiempo t = new Tiempo(1, 20, 30) donde los parámetros que se le
 *    pasan al constructor son las horas, los minutos y los segundos
 *    respectivamente. Crea el método toString para ver los intervalos
 *    de tiempo de la forma 10h 35m 5s. Si se suman por ejemplo 30m 40s
 *    y 35m 20s el resultado debería ser 1h 6m 0s.
 *    Realiza un programa de prueba para comprobar que la clase funciona
 *    bien.
 *
 * @author Luis José Sánchez
 */
public class PruebaTiempo {
  public static void main(String[] args) {
    Tiempo t1 = new Tiempo(1, 20, 30);
    Tiempo t2 = new Tiempo(0, 30, 40);
    Tiempo t3 = new Tiempo(0, 35, 20);

    System.out.println(t1 + " + " + t2 + " = " + t1.suma(t2));
    System.out.println(t2 + " - " + t3 + " = " + t2.resta(t3));
```

```
  }
}
```

Ejercicio 7

Fichero: Zona.java

```
/**
 * Definición de la clase Zona
 *
 * @author Luis José Sánchez
 */
public class Zona {

  private int entradasPorVender;

  public Zona(int n){
    entradasPorVender = n;
  }

  public int getEntradasPorVender() {
    return entradasPorVender;
  }

  /**
   * Vende un número de entradas.
   * <p>
   * Comprueba si quedan entradas libres antes de realizar la venta.
   *
   * @param n número de entradas a vender
   */
  public void vender(int n) {

    if (this.entradasPorVender == 0) {
      System.out.println("Lo siento, las entradas para esa zona están agotadas.");
    } else if (this.entradasPorVender < n) {
      System.out.println("Sólo me quedan " + this.entradasPorVender
                          + " entradas para esa zona.");
    }

    if (this.entradasPorVender >= n) {
      entradasPorVender -= n;
      System.out.println("Aquí tiene sus " + n + " entradas, gracias.");
    }
  }
}
```

Fichero: ExpocochesCampanillas.java

```
/**
 * 7. Queremos gestionar la venta de entradas (no numeradas) de
 *    Expocoches Campanillas que tiene 3 zonas, la sala principal con
 *    1000 entradas disponibles, la zona de compra-venta con 200
 *    entradas disponibles y la zona vip con 25 entradas disponibles.
 *    Hay que controlar que existen entradas antes de venderlas.
 *    La clase <code>Zona</code> con sus atributos y métodos se
 *    proporciona al alumno.
 *    <p>
 *    El menú del programa debe ser el que se muestra a continuación.
 *    Cuando elegimos la opción <code>2</code>, se nos debe preguntar
 *    para qué zona queremos las entradas y cuántas queremos.
 *    Lógicamente, el programa debe controlar que no se puedan vender
 *    más entradas de la cuenta.
 *    <p>
 *    <code>
 *    <pre>
 *    1. Mostrar número de entradas libres
 *    2. Vender entradas
 *    3. Salir
 *    </pre>
 *    </code>
 * @author Luis José Sánchez
 */
public class ExpocochesCampanillas {
  public static void main(String[] args) {

    Zona principal = new Zona(1000);
    Zona compraVenta = new Zona(200);
    Zona vip = new Zona(25);

    int opcion = 0;
    int opcion2 = 0;
    int n = 0;

    do {
      System.out.println("\n\nEXPOCOCHES CAMPANILLAS");
      System.out.println("1. Mostrar número de entradas libres");
      System.out.println("2. Vender entradas");
      System.out.println("3. Salir");
      System.out.println("Elige una opción: ");

      opcion = Integer.parseInt(System.console().readLine());
```

```
      if (opcion == 1) {
        System.out.println("\n\nEn la zona principal hay " + principal.getEntradasPorVender());
        System.out.println("En la zona de compra venta hay " + compraVenta.getEntradasPorVende\
r());
        System.out.println("En la zona vip hay " + vip.getEntradasPorVender());
      }

      if (opcion == 2) {
        System.out.println("\n\n1. Principal");
        System.out.println("2. Compra-venta");
        System.out.println("3. Vip");
        System.out.print("Elige la zona para la que quieres comprar las entradas: ");

        opcion2 = Integer.parseInt(System.console().readLine());

        System.out.print("¿Cuántas entradas quieres? ");
        n = Integer.parseInt(System.console().readLine());

        switch (opcion2) {
          case 1:
            principal.vender(n);
            break;
          case 2:
            compraVenta.vender(n);
            break;
          case 3:
            vip.vender(n);
            break;
          default:
        }
      }
    } while (opcion < 3); // menú principal
  }
}
```

Ejercicio 8

Fichero: Terminal.java

```
class Terminal {

  private String numero;
  private int tiempoDeConversacion; // tiempo de conversación en segundos

  Terminal(String numero) {
    this.numero = numero;
    this.tiempoDeConversacion = 0;
  }

  public String getNumero() {
    return numero;
  }

  public void setNumero(String numero) {
    this.numero = numero;
  }

  public int getTiempoDeConversacion() {
    return tiempoDeConversacion;
  }

  public void setTiempoDeConversacion(int tiempoDeConversacion) {
    this.tiempoDeConversacion = tiempoDeConversacion;
  }

  @Override
  public String toString() {
    return "Nº " + numero + " - " + this.tiempoDeConversacion + "s de conversación ";
  }

  public void llama(Terminal t, int segundosDeLlamada) {
    this.tiempoDeConversacion += segundosDeLlamada;
    t.tiempoDeConversacion += segundosDeLlamada;
  }

}
```

Fichero: PruebaTerminales.java

```java
public class PruebaTerminales {

  public static void main(String[] args) {
    Terminal t1 = new Terminal("678 11 22 33");
    Terminal t2 = new Terminal("644 74 44 69");
    Terminal t3 = new Terminal("622 32 89 09");
    Terminal t4 = new Terminal("664 73 98 18");
    System.out.println(t1);
    System.out.println(t2);
    t1.llama(t2, 320);
    t1.llama(t3, 200);
    System.out.println(t1);
    System.out.println(t2);
    System.out.println(t3);
    System.out.println(t4);
  }

}
```

Ejercicio 9

Fichero: Terminal.java

```java
class Terminal {

  private String numero;
  private int tiempoDeConversacion; // tiempo de conversación en segundos

  Terminal(String numero) {
    this.numero = numero;
    this.tiempoDeConversacion = 0;
  }

  public String getNumero() {
    return numero;
  }

  public void setNumero(String numero) {
    this.numero = numero;
  }

  public int getTiempoDeConversacion() {
    return tiempoDeConversacion;
  }
```

```
  public void setTiempoDeConversacion(int tiempoDeConversacion) {
    this.tiempoDeConversacion = tiempoDeConversacion;
  }

  @Override
  public String toString() {
    return "Nº " + numero + " - " + this.tiempoDeConversacion + "s de conversación ";
  }

  public void llama(Terminal t, int segundosDeLlamada) {
    this.tiempoDeConversacion += segundosDeLlamada;
    t.tiempoDeConversacion += segundosDeLlamada;
  }

}
```

Fichero: PruebaMoviles.java

```
public class PruebaMoviles {

  public static void main(String[] args) {
    Movil m1 = new Movil("678 11 22 33", "rata");
    Movil m2 = new Movil("644 74 44 69", "mono");
    Movil m3 = new Movil("622 32 89 09", "bisonte");
    System.out.println(m1);
    System.out.println(m2);
    m1.llama(m2, 320);
    m1.llama(m3, 200);
    m2.llama(m3, 550);
    System.out.println(m1);
    System.out.println(m2);
    System.out.println(m3);
  }

}
```

Fichero: Movil.java

```
import java.text.DecimalFormat;

public class Movil extends Terminal {

  private String tarifa;
  private double totalTarificado;

  public Movil(String numero, String tarifa) {
    super(numero);
    this.tarifa = tarifa;
    this.totalTarificado = 0;
  }

  @Override
  public void llama(Terminal t, int segundosDeLlamada) {
    super.llama(t, segundosDeLlamada);
    double minutos = (double) segundosDeLlamada / 60;

    switch (this.tarifa) {
      case "rata":
        this.totalTarificado += minutos * 0.06;
        break;
      case "mono":
        this.totalTarificado += minutos * 0.12;
        break;
      case "bisonte":
        this.totalTarificado += minutos * 0.30;
        break;
      default:
    }
  }

  @Override
  public String toString() {
    DecimalFormat formatoEuros = new DecimalFormat("0.00");
    return "Nº " + this.getNumero()
            + " - " + this.getTiempoDeConversacion()
            + "s de conversación - tarificados "
            + formatoEuros.format(this.totalTarificado) + " euros";
  }

}
```

Ejercicio 10

Fichero: Ameba.java

```
class Ameba {
  int peso; // peso en microgramos

  public Ameba() {
    this.peso = 3;
  }

  void come(int pesoComida) {
    this.peso += pesoComida - 1;
  }

  void come(Ameba a) {
    this.peso += a.peso - 1;
    a.peso = 0; // la ameba comida se queda sin sustancia
  }

  @Override
  public String toString() {
    return "Soy una ameba y peso " + peso + " microgramos.";
  }

}
```

Fichero: LaFascinanteVidaDeLasAmebas.java

```
public class LaFascinanteVidaDeLasAmebas {

  public static void main(String[] args) {
    Ameba a1 = new Ameba();
    a1.come(2);
    System.out.println(a1);
    Ameba a2 = new Ameba();
    a2.come(4);
    System.out.println(a2);
    a1.come(a2);
    System.out.println(a1);
    System.out.println(a2);
    a2.come(3);
    System.out.println(a2);
  }

}
```

Ejercicio 11

Fichero: ElCorteIslandes.java

```
public class ElCorteIslandes {
  public static void main(String[] args) {
    TarjetaRegalo t1 = new TarjetaRegalo(100);
    TarjetaRegalo t2 = new TarjetaRegalo(120);
    System.out.println(t1);
    System.out.println(t2);
    t1.gasta(45.90);
    t2.gasta(5);
    t2.gasta(200);
    t1.gasta(3.55);
    System.out.println(t1);
    System.out.println(t2);
    TarjetaRegalo t3 = t1.fusionaCon(t2);
    System.out.println(t1);
    System.out.println(t2);
    System.out.println(t3);
  }
}
```

Fichero: TarjetaRegalo.java

```
import java.text.DecimalFormat;

public class TarjetaRegalo {
  String numero;
  double saldo;

  public TarjetaRegalo(double saldo) {
    this.saldo = saldo;

    // Genera un número de tarjeta aleatorio de 5 cifras
    this.numero = "";
    for (int i = 0; i < 5; i++) {
      this.numero += (int)(Math.random() * 10);
    }
  }

  @Override
  public String toString() {
    DecimalFormat dosDecimales = new DecimalFormat("0.00");
    return "Tarjeta nº " + numero + " - Saldo " + dosDecimales.format(saldo) + "€";
  }

  void gasta(double gasto) {
    if (gasto > saldo) {
      System.out.printf("No tiene suficiente saldo para gastar %.2f€\n", gasto);
```

```
    } else {
      saldo -= gasto;
    }
  }

  TarjetaRegalo fusionaCon(TarjetaRegalo t) {
    double nuevoSaldo = this.getSaldo() + t.getSaldo();
    this.setSaldo(0);
    t.setSaldo(0);
    return new TarjetaRegalo(nuevoSaldo);
  }

  public void setSaldo(double saldo) {
    this.saldo = saldo;
  }

  public double getSaldo() {
    return saldo;
  }
}
```

Ejercicio 12

Fichero: Biblioteca.java

```
public class Biblioteca {

  public static void main(String[] args) {
    Libro libro1 = new Libro("123456", "La Ruta Prohibida", 2007);
    Libro libro2 = new Libro("112233", "Los Otros", 2016);
    Libro libro3 = new Libro("456789", "La rosa del mundo", 1995);
    Revista revista1 = new Revista("444555", "Año Cero", 2019, 344);
    Revista revista2 = new Revista("002244", "National Geographic", 2003, 255);
    System.out.println(libro1);
    System.out.println(libro2);
    System.out.println(libro3);
    System.out.println(revista1);
    System.out.println(revista2);
    libro2.presta();
    if (libro2.estaPrestado()) {
      System.out.println("El libro está prestado");
    }
    libro2.presta();
    libro2.devuelve();
    if (libro2.estaPrestado()) {
```

```
      System.out.println("El libro está prestado");
    }
    libro3.presta();
    System.out.println(libro2);
    System.out.println(libro3);
  }

}
```

Fichero: Publicacion.java

```
public class Publicacion {
  private String isbn;
  private String titulo;
  private int anio;

  public Publicacion(String isbn, String titulo, int anio) {
    this.isbn = isbn;
    this.titulo = titulo;
    this.anio = anio;
  }

  @Override
  public String toString() {
    return "ISBN: " + isbn + ", título: " + titulo + ", año de publicación: " + anio;
  }
}
```

Fichero: Revista.java

```
public class Revista extends Publicacion {
  private int numero;

  public Revista(String isbn, String titulo, int anio, int numero) {
    super(isbn, titulo, anio);
    this.numero = numero;
  }
}
```

Fichero: Prestable.java

```
public interface Prestable {
  public void presta();
  public void devuelve();
  public boolean estaPrestado();
}
```

Fichero: Libro.java

```
public class Libro extends Publicacion implements Prestable {
  private boolean prestado = false;

  public Libro(String isbn, String titulo, int anio) {
    super(isbn, titulo, anio);
  }

  @Override
  public void presta() {
    if (this.prestado) {
      System.out.println("Lo siento, ese libro ya está prestado.");
    } else {
      this.prestado = true;
    }
  }

  @Override
  public void devuelve() {
    this.prestado = false;
  }

  @Override
  public String toString() {
    return super.toString() + " (" + (this.prestado ? "prestado" : "no prestado") + ")";
  }

  @Override
  public boolean estaPrestado() {
    return this.prestado;
  }
}
```

Ejercicio 13

Fichero: Banco.java

```
public class Banco {

  public static void main(String[] args) {
    CuentaCorriente cuenta1 = new CuentaCorriente();
    CuentaCorriente cuenta2 = new CuentaCorriente(1500);
    CuentaCorriente cuenta3 = new CuentaCorriente(6000);
    System.out.println(cuenta1);
    System.out.println(cuenta2);
    System.out.println(cuenta3);
    cuenta1.ingreso(2000);
    cuenta2.cargo(600);
    cuenta3.ingreso(75);
    cuenta1.cargo(55);
    cuenta2.transferencia(cuenta3, 100);
    System.out.println(cuenta1);
    System.out.println(cuenta2);
    System.out.println(cuenta3);
  }

}
```

Fichero: CuentaCorriente.java

```
class CuentaCorriente {

  private String numero = "";
  private double saldo;

  public CuentaCorriente() {
    this.generaNumero();
  }

  public CuentaCorriente(double saldo) {
    this.generaNumero();
    this.saldo = saldo;
  }

  private void generaNumero() {
    for (int i = 0; i < 10; i++) {
      numero += (int) (Math.random() * 10);
    }
  }

  @Override
  public String toString() {
    return "Número de cta: " + numero + " Saldo: " + String.format("%.2f", saldo) + " €";
```

```
  }

  void ingreso(int dinero) {
    saldo += dinero;
  }

  void cargo(int dinero) {
    saldo -= dinero;
  }

  void transferencia(CuentaCorriente cuentaDestino, int dinero) {
    saldo -= dinero;
    cuentaDestino.saldo += dinero;
  }
}
```

Ejercicio 14

Fichero: S92Ejercicio14.java

```
public class S92Ejercicio14 {

  public static void main(String[] args) {
    FichaDomino f1 = new FichaDomino(6, 1);
    FichaDomino f2 = new FichaDomino(0, 4);
    FichaDomino f3 = new FichaDomino(3, 3);
    FichaDomino f4 = new FichaDomino(0, 1);
    System.out.println(f1);
    System.out.println(f2);
    System.out.println(f3);
    System.out.println(f4);

    System.out.println(f2.voltea());
    System.out.println(f2.encaja(f4));
    System.out.println(f1.encaja(f4));
    System.out.println(f1.encaja(f3));
    System.out.println(f1.encaja(f2));
  }

}
```

Fichero: FichaDomino.java

```
public class FichaDomino {
  private int lado1;
  private int lado2;

  public FichaDomino(int lado1, int lado2) {
    this.lado1 = lado1;
    this.lado2 = lado2;
  }

  FichaDomino voltea() {
    return new FichaDomino(lado2, lado1);
  }

  boolean encaja(FichaDomino ficha) {
    return (lado1 == ficha.lado1) || (lado1 == ficha.lado2) ||
           (lado2 == ficha.lado1) || (lado2 == ficha.lado2);
  }

  @Override
  public String toString() {
    return "[" + lado1 + " | " + (lado2 == 0 ? " " : lado2) + ']';
  }
}
```

Ejercicio 15

Fichero: S92Ejercicio15.java

```
public class S92Ejercicio15 {

  public static void main(String[] args) {
    int cuentaFichas = 0;

    FichaDomino[] fichas = new FichaDomino[8];

    fichas[0] = new FichaDomino();

    for (int i = 1; i < 8; i++) {
      FichaDomino aux;
      do {
        aux = new FichaDomino();
      } while (!fichas[i - 1].encajaEnFila(aux));
      fichas[i] = aux;
    }
```

```
    for (FichaDomino ficha : fichas) {
      System.out.print(ficha);
    }
  }

}
```

Fichero: FichaDomino.java

```
public class FichaDomino {

  private int lado1;
  private int lado2;

  public FichaDomino() {
    this.lado1 = (int) (Math.random() * 7);
    this.lado2 = (int) (Math.random() * 7);
  }

  public FichaDomino(int lado1, int lado2) {
    this.lado1 = lado1;
    this.lado2 = lado2;
  }

  FichaDomino voltea() {
    return new FichaDomino(lado2, lado1);
  }

  boolean encaja(FichaDomino ficha) {
    return (lado1 == ficha.lado1) || (lado1 == ficha.lado2)
            || (lado2 == ficha.lado1) || (lado2 == ficha.lado2);
  }

  boolean encajaEnFila(FichaDomino ficha) {
    return lado2 == ficha.lado1;
  }

  @Override
  public String toString() {
    return "[" + (lado1 == 0 ? " " : lado1) + "|" + (lado2 == 0 ? " " : lado2) + ']';
  }

}
```

Ejercicio 16

Fichero: S92Ejercicio16.java

```
public class S92Ejercicio16 {

  public static void main(String[] args) {
    Punto p1 = new Punto(4.21, 7.3);
    Punto p2 = new Punto(-2, 1.66);
    Linea l = new Linea(p1, p2);
    System.out.println(l);
  }

}
```

Fichero: Punto.java

```
public class Punto {

  private double x;
  private double y;

  public Punto(double x, double y) {
    this.x = x;
    this.y = y;
  }

  @Override
  public String toString() {
    return "(" + x + ", " + y + ")";
  }

}
```

Fichero: Linea.java

```
class Linea {

  private Punto origen;
  private Punto fin;

  public Linea(Punto origen, Punto fin) {
    this.origen = origen;
    this.fin = fin;
  }

  @Override
  public String toString() {
    return "Línea formada por los puntos " + origen + " y " + fin;
  }

}
```

Ejercicio 17

Fichero: S92Ejercicio17.java

```
public class S92Ejercicio17 {

  public static void main(String[] args) {
    Piramide p = new Piramide(4);
    Rectangulo r1 = new Rectangulo(4, 3);
    Rectangulo r2 = new Rectangulo(6, 2);
    System.out.println(p);
    System.out.println(r1);
    System.out.println(r2);
    System.out.println("Pirámides creadas: " + Piramide.getPiramidesCreadas()
    );
    System.out.println("Rectángulos creados: " + Rectangulo.getRectangulosCreados());
  }

}
```

Fichero: Rectangulo.java

```
class Rectangulo {
  private int base;
  private int altura;
  private static int rectangulosCreados;

  public Rectangulo(int base, int altura) {
    this.base = base;
    this.altura = altura;
    rectangulosCreados++;
  }

  public static int getRectangulosCreados() {
    return rectangulosCreados;
  }

  @Override
  public String toString() {
    String cadena = "";

    for (int i = 0; i < altura; i++) {
      for (int j = 0; j < base; j++) {
        cadena += '*';
      }
      cadena += '\n';
    }
    return cadena;
  }

}
```

Fichero: Piramide.java

```
class Piramide {
  private int altura;
  private static int piramidesCreadas;

  public Piramide(int altura) {
    this.altura = altura;
    piramidesCreadas++;
  }

  public static int getPiramidesCreadas() {
    return piramidesCreadas;
  }

  @Override
```

```
  public String toString() {
    String cadena = "";
    int planta = 1;
    int longitudDeLinea = 1;
    int espacios = altura - 1;

    while (planta <= altura) {

      // inserta espacios
      for (int i = 1; i <= espacios; i++) {
        cadena += ' ';
      }

      // pinta la línea
      for (int i = 1; i <= longitudDeLinea; i++) {
        cadena += '*';
      }

      cadena += '\n';

      planta++;
      espacios--;
      longitudDeLinea += 2;
    }
    return cadena;
  }

}
```

Ejercicio 18

Fichero: S92Ejercicio18.java

```
public class S92Ejercicio18 {

  public static void main(String[] args) {
    Incidencia inc1 = new Incidencia(105, "No tiene acceso a internet");
    Incidencia inc2 = new Incidencia(14, "No arranca");
    Incidencia inc3 = new Incidencia(5, "La pantalla se ve rosa");
    Incidencia inc4 = new Incidencia(237, "Hace un ruido extraño");
    Incidencia inc5 = new Incidencia(111, "Se cuelga al abrir 3 ventanas");
    inc2.resuelve("El equipo no estaba enchufado");
    inc3.resuelve("Cambio del cable VGA");
    System.out.println(inc1);
    System.out.println(inc2);
```

```
    System.out.println(inc3);
    System.out.println(inc4);
    System.out.println(inc5);
    System.out.println("Incidencias pendientes: " + Incidencia.getPendientes());
  }

}
```

Fichero: Incidencia.java

```
class Incidencia {

  private int codigo;
  private String estado;
  private int puesto;
  private String problema;
  private String resolucion;

  private static int contadorCodigos = 1;
  private static int pendientes = 0;

  public Incidencia(int puesto, String problema) {
    this.puesto = puesto;
    this.problema = problema;
    this.estado = "Pendiente";
    this.codigo = contadorCodigos++;
    pendientes++;
  }

  void resuelve(String resolucion) {
    this.resolucion = resolucion;
    this.estado = "Resuelta";
    pendientes--;
  }

  @Override
  public String toString() {
    return "Incidencia " + codigo + " - Puesto: " + puesto + " - " + problema
      + " - " + estado + (estado.equals("Resuelta") ? " - " + resolucion : "");
  }

  static int getPendientes() {
    return pendientes;
  }
}
```

Arrays de objetos

Ejercicio 1

Fichero: ArrayDeGatos.java

```
/**
 * 1. Utiliza la clase Gato para crear un array de cuatro gatos e
 *    introduce los datos de cada uno de ellos mediante un bucle.
 *    Muestra a continuación los datos de todos los gatos utilizando
 *    también un bucle.
 *
 * @author Luis José Sánchez
 */
public class ArrayDeGatos {
  public static void main(String[] args) {

    Gato[] gato = new Gato[4];

    int i;

    System.out.println("\nPor favor, introduzca los datos de los gatos.");

    for (i = 0; i < 4; i++) {
      gato[i] = new Gato();
      System.out.println("  \nGato nº " + (i + 1));
      System.out.print("Nombre: ");
      gato[i].setNombre(System.console().readLine());
      System.out.print("Color: ");
      gato[i].setColor(System.console().readLine());
      System.out.print("Raza: ");
      gato[i].setRaza(System.console().readLine());
    }

    System.out.println("\n\nDatos de los gatos.");

    for (i = 0; i < 4; i++) {
      System.out.println("\nGato nº" + (i + 1));
      System.out.println("Nombre: " + gato[i].getNombre());
      System.out.println("Color: " + gato[i].getColor());
      System.out.println("Raza: " + gato[i].getRaza());
    }
  }
}
```

Fichero: Gato.java

```
public class Gato {
  private String nombre;
  private String color;
  private String raza;

  public String getNombre() {
    return nombre;
  }

  public void setNombre(String nombre) {
    this.nombre = nombre;
  }

  public String getColor() {
    return color;
  }

  public void setColor(String color) {
    this.color = color;
  }

  public String getRaza() {
    return raza;
  }

  public void setRaza(String raza) {
    this.raza = raza;
  }
}
```

Ejercicio 2

Fichero: ArrayDeGatos02.java

```
/**
 * 2. Cambia el programa anterior de tal forma que los datos de los
 *    gatos se introduzcan directamente en el código de la forma
 *    <code>gatito[2].setColor("marrón")</code> o bien mediante el
 *    constructor, de la forma <code>gatito[3] = new Gato("Garfeld",
 *    "naranja", "macho")</code>.
 *    <p>
 *    Muestra a continuación los datos de todos los gatos utilizando un
 *    bucle.
 *
 * @author Luis José Sánchez
```

```
 */
public class ArrayDeGatos02 {
  public static void main(String[] args) {
    Gato[] gato = new Gato[4];

    int i;

    gato[0] = new Gato("Garfield", "naranja", "mestizo");
    gato[1] = new Gato("Pepe", "gris", "angora");
    gato[2] = new Gato("Mauri", "blanco", "manx");
    gato[3] = new Gato("Ulises", "marrón", "persa");

    System.out.println("\nDatos de los gatos.");

    for (i = 0; i < 4; i++) {
      System.out.println("\nGato nº" + (i + 1));
      System.out.println("Nombre: " + gato[i].getNombre());
      System.out.println("Color: " + gato[i].getColor());
      System.out.println("Raza: " + gato[i].getRaza());
    }
  }
}
```

Fichero: Gato.java

```
public class Gato {
  private String nombre;
  private String color;
  private String raza;

  public Gato(String nombre, String color, String raza) {
    this.nombre = nombre;
    this.color = color;
    this.raza = raza;
  }

  public String getNombre() {
    return nombre;
  }
  public void setNombre(String nombre) {
    this.nombre = nombre;
  }

  public String getColor() {
    return color;
  }
```

```
  public void setColor(String color) {
    this.color = color;
  }

  public String getRaza() {
    return raza;
  }

  public void setRaza(String raza) {
    this.raza = raza;
  }
}
```

Ejercicio 3

Fichero: ColeccionDeDiscosPrincipal.java

```
/**
 * ColeccionDeDiscosPrincipal.java
 * Gestión de una colección de discos.
 * @author Luis José Sánchez
 */
public class ColeccionDeDiscosPrincipal {

  // N determina el tamaño del array
  static int N = 100;

  public static void main(String[] args) {

    //Crea el array de discos
    Disco[] album = new Disco[N];

    // Crea todos los discos que van en cada una de
    // las celdas del array
    for(int i = 0; i < N; i++) {
      album[i] = new Disco();
    }

    int opcion;
    String codigoIntroducido;
    String autorIntroducido;
    String tituloIntroducido;
    String generoIntroducido;
    String duracionIntroducidaString;
```

```
    int duracionIntroducida;
    int primeraLibre;
    int i;

    do {
      System.out.println("\n\nCOLECCIÓN DE DISCOS");
      System.out.println("===================");
      System.out.println("1. Listado");
      System.out.println("2. Nuevo disco");
      System.out.println("3. Modificar");
      System.out.println("4. Borrar");
      System.out.println("5. Salir");
      System.out.print("Introduzca una opción: ");
      opcion = Integer.parseInt(System.console().readLine());

      switch (opcion) {
      case 1:
        System.out.println("\nLISTADO");
        System.out.println("=======");
        for(i = 0; i < N; i++) {
          if (!album[i].getCodigo().equals("LIBRE")) {
            System.out.println(album[i]);
          }
        }
        break;

      case 2:
        System.out.println("\nNUEVO DISCO");
        System.out.println("===========");

        // Busca la primera posición libre del array
        primeraLibre = -1;
        do {
          primeraLibre++;
        } while (!((album[primeraLibre].getCodigo()).equals("LIBRE")));

        System.out.println("Por favor, introduzca los datos del disco.");

        System.out.print("Código: ");
        codigoIntroducido = System.console().readLine();
        album[primeraLibre].setCodigo(codigoIntroducido);

        System.out.print("Autor: ");
        autorIntroducido = System.console().readLine();
        album[primeraLibre].setAutor(autorIntroducido);
```

```
        System.out.print("Título: ");
        tituloIntroducido = System.console().readLine();
        album[primeraLibre].setTitulo(tituloIntroducido);

        System.out.print("Género: ");
        generoIntroducido = System.console().readLine();
        album[primeraLibre].setGenero(generoIntroducido);

        System.out.print("Duración: ");
        duracionIntroducida = Integer.parseInt(System.console().readLine());
        album[primeraLibre].setDuracion(duracionIntroducida);

        break;

      case 3:
        System.out.println("\nMODIFICAR");
        System.out.println("===========");

        System.out.print("Por favor, introduzca el código del disco cuyos datos desea cambiar:\
 ");
        codigoIntroducido = System.console().readLine();

        i = -1;
        do {
          i++;
        } while (!((album[i].getCodigo()).equals(codigoIntroducido)));

        System.out.println("Introduzca los nuevos datos del disco o INTRO para dejarlos igual.\
");

        System.out.println("Código: " + album[i].getCodigo());
        System.out.print("Nuevo código: ");
        codigoIntroducido = System.console().readLine();
        if (!codigoIntroducido.equals("")) {
          album[i].setCodigo(codigoIntroducido);
        }

        System.out.println("Autor: " + album[i].getAutor());
        System.out.print("Nuevo autor: ");
        autorIntroducido = System.console().readLine();
        if (!autorIntroducido.equals("")) {
          album[i].setAutor(autorIntroducido);
        }

        System.out.println("Título: " + album[i].getTitulo());
        System.out.print("Nuevo título: ");
```

```
        tituloIntroducido = System.console().readLine();
        if (!tituloIntroducido.equals("")) {
          album[i].setTitulo(tituloIntroducido);
        }

        System.out.println("Género: " + album[i].getGenero());
        System.out.print("Nuevo género: ");
        generoIntroducido = System.console().readLine();
        if (!generoIntroducido.equals("")) {
          album[i].setGenero(generoIntroducido);
        }

        System.out.println("Duración: " + album[i].getDuracion());
        System.out.print("Duración: ");
        duracionIntroducidaString = System.console().readLine();
        if (!duracionIntroducidaString.equals("")) {
          album[i].setDuracion(Integer.parseInt(duracionIntroducidaString));
        }

        break;

      case 4:
        System.out.println("\nBORRAR");
        System.out.println("======");

        System.out.print("Por favor, introduzca el código del disco que desea borrar: ");
        codigoIntroducido = System.console().readLine();

        i = -1;
        do {
          i++;
        } while (!((album[i].getCodigo()).equals(codigoIntroducido)));
        album[i].setCodigo("LIBRE");
        System.out.println("Album borrado.");

        break;

      default:

      } // switch
    } while (opcion != 5);
  }
}
```

Fichero: Disco.java

```
/**
 * Definición de la clase Disco
 *
 * @author Luis José Sánchez
 */

public class Disco {
  private String codigo = "LIBRE";
  private String autor;
  private String titulo;
  private String genero;
  private int duracion; // duración total en minutos

  public String getCodigo() {
    return codigo;
  }

  public void setCodigo(String codigo) {
    this.codigo = codigo;
  }

  public String getAutor() {
    return autor;
  }

  public void setAutor(String autor) {
    this.autor = autor;
  }

  public String getGenero() {
    return genero;
  }

  public void setGenero(String genero) {
    this.genero = genero;
  }

  public String getTitulo() {
    return titulo;
  }

  public void setTitulo(String titulo) {
    this.titulo = titulo;
  }

  public int getDuracion() {
```

```
    return duracion;
  }

  public void setDuracion(int duracion) {
    this.duracion = duracion;
  }

  public String toString() {
    String cadena = "\n---------------------------------------";
        cadena += "\nCódigo: " + this.codigo;
        cadena += "\nAutor: " + this.autor;
        cadena += "\nTítulo: " + this.titulo;
        cadena += "\nGénero: " + this.genero;
        cadena += "\nDuración: " + this.duracion;
        cadena += "\n---------------------------------------";

    return cadena;
  }
}
```

Ejercicio 4

Fichero: ColeccionDeDiscosMejorado.java

```
/*
 * 4. Modifca el programa "Colección de discos" como se indica a continuación:
 *     a) Mejora la opción "Nuevo disco" de tal forma que cuando se llenen todas
 *        las posiciones del array, el programa muestre un mensaje de error. No se
 *        permitirá introducir los datos de ningún disco hasta que no se borre alguno
 *        de la lista.
 *     b) Mejora la opción "Borrar" de tal forma que se verifique que el código
 *        introducido por el usuario existe.
 *     c) Modifca el programa de tal forma que el código del disco sea único, es
 *        decir que no se pueda repetir.
 *     d) Crea un submenú dentro dentro de "Listado" de tal forma que exista un
 *        listado completo, un listado por autor (todos los discos que ha publicado un
 *        determinado autor), un listado por género (todos los discos de un género
 *        determinado) y un listado de discos cuya duración esté en un rango
 *        determinado por el usuario.
 *
 * @author Luis José Sánchez
 */

public class ColeccionDeDiscosMejorado {
```

```
// N determina el tamaño del array
static int N = 100;

public static void main(String[] args) {

  int i;
  String codigoIntroducido = "";
  String codigo;
  String autorIntroducido;
  String tituloIntroducido;
  String generoIntroducido;
  String duracionIntroducidaString;
  int opcion;
  int opcionListado;
  int duracionIntroducida;
  int primeraLibre = -1;
  int limiteInferior;
  int limiteSuperior;
  boolean existeCodigo = true;

  //Crea el array de discos
  Disco[] album = new Disco[N];

  // Crea todos los discos que van en cada una de
  // las celdas del array
  for(i = 0; i < N; i++) {
    album[i] = new Disco();
  }

  // Menu
  do {
    System.out.println("\n\nCOLECCIÓN DE DISCOS");
    System.out.println("===================");
    System.out.println("1. Listado");
    System.out.println("2. Nuevo disco");
    System.out.println("3. Modificar");
    System.out.println("4. Borrar");
    System.out.println("5. Salir");
    System.out.print("Introduzca una opción: ");
    opcion = Integer.parseInt(System.console().readLine());

    switch (opcion) {

    //////////////////////////////////////////////////////////////////////////////
    // LISTADO ///////////////////////////////////////////////////////////////////
    //////////////////////////////////////////////////////////////////////////////
```

```
      case 1:
        do {
          System.out.println("\nLISTADO");
          System.out.println("=======");
          System.out.println("1. Completo");
          System.out.println("2. Por autor");
          System.out.println("3. Por género");
          System.out.println("4. En un rango de duración");
          System.out.println("5. Menú principal");
          System.out.print("Introduzca una opción: ");
          opcionListado = Integer.parseInt(System.console().readLine());

          switch (opcionListado) {

            case 1: // Listado completo ////////////////

              for(i = 0; i < N; i++) {
                if (!album[i].getCodigo().equals("LIBRE")) {
                  System.out.println("------------------------------------------");
                  System.out.println("Código: " + album[i].getCodigo());
                  System.out.println("Autor: " + album[i].getAutor());
                  System.out.println("Título: " + album[i].getTitulo());
                  System.out.println("Género: " + album[i].getGenero());
                  System.out.println("Duración: " + album[i].getDuracion());
                  System.out.println("------------------------------------------");
                }
              }
              break;

            case 2: // Listado por autor ////////////////

              System.out.print("Introduzca el autor: ");
              autorIntroducido = System.console().readLine();

              for(i = 0; i < N; i++) {
                if ( (!album[i].getCodigo().equals("LIBRE")) && (album[i].getAutor().equals(au\
torIntroducido)) ){
                  System.out.println("------------------------------------------");
                  System.out.println("Código: " + album[i].getCodigo());
                  System.out.println("Autor: " + album[i].getAutor());
                  System.out.println("Título: " + album[i].getTitulo());
                  System.out.println("Género: " + album[i].getGenero());
                  System.out.println("Duración: " + album[i].getDuracion());
                  System.out.println("------------------------------------------");
                }
```

```
          }
          break;

        case 3: // Listado por género ////////////////

          System.out.print("Introduzca el género: ");
          generoIntroducido = System.console().readLine();

          for(i = 0; i < N; i++) {
            if ( (!album[i].getCodigo().equals("LIBRE")) && (album[i].getGenero().equals(g\
eneroIntroducido)) ){
              System.out.println("------------------------------------------");
              System.out.println("Código: " + album[i].getCodigo());
              System.out.println("Autor: " + album[i].getAutor());
              System.out.println("Título: " + album[i].getTitulo());
              System.out.println("Género: " + album[i].getGenero());
              System.out.println("Duración: " + album[i].getDuracion());
              System.out.println("------------------------------------------");
            }
          }
          break;

        case 4: // Listado en un rango de duración ////////////////

          System.out.println("Establezca el intervalo para la duración");
          System.out.print("Introduzca el límite inferior de duración en minutos: ");
          limiteInferior = Integer.parseInt(System.console().readLine());
          System.out.print("Introduzca el límite superior de duración en minutos: ");
          limiteSuperior = Integer.parseInt(System.console().readLine());

          for(i = 0; i < N; i++) {
            if ( (!album[i].getCodigo().equals("LIBRE")) && (album[i].getDuracion() <= lim\
iteSuperior) && (album[i].getDuracion() >= limiteInferior) ){
              System.out.println("------------------------------------------");
              System.out.println("Código: " + album[i].getCodigo());
              System.out.println("Autor: " + album[i].getAutor());
              System.out.println("Título: " + album[i].getTitulo());
              System.out.println("Género: " + album[i].getGenero());
              System.out.println("Duración: " + album[i].getDuracion());
              System.out.println("------------------------------------------");
            }
          }

      } // switch (opcionListado)
    } while (opcionListado != 5);
```

```
    break;

  //////////////////////////////////////////////////////////////////////////////
  // NUEVO DISCO ///////////////////////////////////////////////////////////////
  //////////////////////////////////////////////////////////////////////////////
  case 2:
    System.out.println("\nNUEVO DISCO");
    System.out.println("===========");

    // Busca la primera posición libre del array
    primeraLibre = 0;
    codigo = album[primeraLibre].getCodigo();
    while ((primeraLibre < N) && (!(codigo.equals("LIBRE")))) {
      primeraLibre++;
      if (primeraLibre < N) {
        codigo = album[primeraLibre].getCodigo();
      }
    }

    if (primeraLibre == N) {
      System.out.println("Lo siento, a base de datos está llena.");
    } else {
      System.out.println("Por favor, introduzca los datos del disco.");
      System.out.print("Código: ");

      // Comprueba que el código introducido no se repita
      existeCodigo = true;
      while (existeCodigo) {
        existeCodigo = false;
        codigoIntroducido = System.console().readLine();

        for (i = 0; i < N; i++)
          if (codigoIntroducido.equals(album[i].getCodigo())) {
            existeCodigo = true;
          }

        if (existeCodigo) {
          System.out.println("Ese código ya existe en la base de datos.");
          System.out.print("Introduzca otro código: ");
        }
      } // while (existeCodigo)

      album[primeraLibre].setCodigo(codigoIntroducido);

      System.out.print("Autor: ");
      autorIntroducido = System.console().readLine();
```

```
          album[primeraLibre].setAutor(autorIntroducido);

          System.out.print("Título: ");
          tituloIntroducido = System.console().readLine();
          album[primeraLibre].setTitulo(tituloIntroducido);

          System.out.print("Género: ");
          generoIntroducido = System.console().readLine();
          album[primeraLibre].setGenero(generoIntroducido);

          System.out.print("Duración: ");
          duracionIntroducida = Integer.parseInt(System.console().readLine());
          album[primeraLibre].setDuracion(duracionIntroducida);
        }

        break;

      //////////////////////////////////////////////////////////////////////////////
      // MODIFICAR /////////////////////////////////////////////////////////////////
      //////////////////////////////////////////////////////////////////////////////

      case 3:
        System.out.println("\nMODIFICAR");
        System.out.println("===========");

        System.out.print("Por favor, introduzca el código del disco cuyos datos desea cambiar:\
 ");
        codigoIntroducido = System.console().readLine();

        i = -1;
        do {
          i++;
        } while (!((album[i].getCodigo()).equals(codigoIntroducido)));

        System.out.println("Introduzca los nuevos datos del disco o INTRO para dejarlos igual.\
");

        System.out.println("Código: " + album[i].getCodigo());
        System.out.print("Nuevo código: ");
        codigoIntroducido = System.console().readLine();
        if (!codigoIntroducido.equals("")) {
          album[i].setCodigo(codigoIntroducido);
        }

        System.out.println("Autor: " + album[i].getAutor());
        System.out.print("Nuevo autor: ");
```

```
    autorIntroducido = System.console().readLine();
    if (!autorIntroducido.equals("")) {
      album[i].setAutor(autorIntroducido);
    }
    System.out.println("Título: " + album[i].getTitulo());
    System.out.print("Nuevo título: ");
    tituloIntroducido = System.console().readLine();
    if (!tituloIntroducido.equals("")) {
      album[i].setTitulo(tituloIntroducido);
    }
    System.out.println("Género: " + album[i].getGenero());
    System.out.print("Nuevo género: ");
    generoIntroducido = System.console().readLine();
    if (!generoIntroducido.equals("")) {
      album[i].setGenero(generoIntroducido);
    }
    System.out.println("Duración: " + album[i].getDuracion());
    System.out.print("Duración: ");
    duracionIntroducidaString = System.console().readLine();
    if (!duracionIntroducidaString.equals("")) {
      album[i].setDuracion(Integer.parseInt(duracionIntroducidaString));
    }
    break;


  //////////////////////////////////////////////////////////////////////////////
  // BORRAR ////////////////////////////////////////////////////////////////////
  //////////////////////////////////////////////////////////////////////////////

  case 4:
    System.out.println("\nBORRAR");
    System.out.println("======");

    System.out.print("Por favor, introduzca el código del disco que desea borrar: ");
    codigoIntroducido = System.console().readLine();

    i = -1;
    codigo = "";
    do {
      System.out.println(i);
      i++;
      if (i < N) {
        codigo = album[i].getCodigo();
      }
    } while (!(codigo.equals(codigoIntroducido)) && (i < N));
```

```
        System.out.println(i);

        if (i == N) {
          System.out.println("Lo siento, el código introducido no existe.");
        } else {
          album[i].setCodigo("LIBRE");
          System.out.println("Album borrado.");
        }

      } // switch
    } while (opcion != 5);
  }
}
```

Fichero: Disco.java

```
/**
 * Definición de la clase Disco
 *
 * @author Luis José Sánchez
 */

public class Disco {
  private String codigo = "LIBRE";
  private String autor;
  private String titulo;
  private String genero;
  private int duracion; // duración total en minutos

  public String getCodigo() {
    return codigo;
  }

  public void setCodigo(String codigo) {
    this.codigo = codigo;
  }

  public String getAutor() {
    return autor;
  }

  public void setAutor(String autor) {
    this.autor = autor;
  }

  public String getGenero() {
```

```
    return genero;
  }

  public void setGenero(String genero) {
    this.genero = genero;
  }

  public String getTitulo() {
    return titulo;
  }

  public void setTitulo(String titulo) {
    this.titulo = titulo;
  }

  public int getDuracion() {
    return duracion;
  }

  public void setDuracion(int duracion) {
    this.duracion = duracion;
  }

  public String toString() {
    String cadena = "\n------------------------------------------";
        cadena += "\nCódigo: " + this.codigo;
        cadena += "\nAutor: " + this.autor;
        cadena += "\nTítulo: " + this.titulo;
        cadena += "\nGénero: " + this.genero;
        cadena += "\nDuración: " + this.duracion;
        cadena += "\n------------------------------------------";

    return cadena;
  }
}
```

Ejercicio 5

Fichero: Articulo.java

```
public class Articulo {
  private String codigo = "LIBRE";
  private String descripcion;
  private double precioDeCompra;
  private double precioDeVenta;
  private int stock;

  public String getCodigo() {
    return codigo;
  }

  public void setCodigo(String codigo) {
    this.codigo = codigo;
  }

  public String getDescripcion() {
    return descripcion;
  }

  public void setDescripcion(String descripcion) {
    this.descripcion = descripcion;
  }

  public double getPrecioDeCompra() {
    return precioDeCompra;
  }

  public void setPrecioDeCompra(double precioDeCompra) {
    this.precioDeCompra = precioDeCompra;
  }

  public double getPrecioDeVenta() {
    return precioDeVenta;
  }

  public void setPrecioDeVenta(double precioDeVenta) {
    this.precioDeVenta = precioDeVenta;
  }

  public int getStock() {
    return stock;
  }

  public void setStock(int stock) {
    this.stock = stock;
  }
```

```
  public String toString() {
    String cadena = "------------------------------------------";
    cadena += "\nCódigo: " + this.codigo;
    cadena += "\nDescripción: " + this.descripcion;
    cadena += "\nPrecio de compra: " + this.precioDeCompra;
    cadena += "\nPrecio de venta: " + this.precioDeVenta;
    cadena += "\nStock: " + this.stock + " unidades";
    cadena += "\n------------------------------------------";
    return cadena;
  }
}
```

Fichero: Gestisimal.java

```
/*
 * 5. Crea el programa GESTISIMAL (GESTIón SIMplifcada de Almacén) para llevar
 *    el control de los artículos de un almacén. De cada artículo se debe saber
 *    el código, la descripción, el precio de compra, el precio de venta y el
 *    stock (número de unidades). El menú del programa debe tener, al menos, las
 *    siguientes opciones:
 *    <p>
 *    1. Listado
 *    2. Alta
 *    3. Baja
 *    4. Modifcación
 *    5. Entrada de mercancía
 *    6. Salida de mercancía
 *    7. Salir
 *    <p>
 *    La entrada y salida de mercancía supone respectivamente el incremento y
 *    decremento de stock de un determinado artículo. Hay que controlar que no
 *    se pueda sacar más mercancía de la que hay en el almacén.
 *
 *    @author Luis José Sánchez
 */

public class Gestisimal {

  // N determina el tamaño del array
  static int N = 100;

  public static void main(String[] args) {

    int opcion;
    int primeraLibre;
```

```
int i;
int stockIntroducido;
double precioDeCompraIntroducido;
double precioDeVentaIntroducido;
String codigo;
String codigoIntroducido = "";
String descripcionIntroducida;
String precioDeCompraIntroducidoString;
String precioDeVentaIntroducidoString;
String stockIntroducidoString;
boolean existeCodigo;

//Crea el array de artículos
Articulo[] articulo = new Articulo[N];

// Crea todos los artículos que van en cada una de
// las celdas del array
for(i = 0; i < N; i++) {
  articulo[i] = new Articulo();
}

// Menu
do {
  System.out.println("\n\nG E S T I S I M A L");
  System.out.println("===================");
  System.out.println("1. Listado");
  System.out.println("2. Alta");
  System.out.println("3. Baja");
  System.out.println("4. Modificación");
  System.out.println("5. Entrada de mercancía");
  System.out.println("6. Salida de mercancía");
  System.out.println("7. Salir");
  System.out.print("Introduzca una opción: ");
  opcion = Integer.parseInt(System.console().readLine());

  switch (opcion) {

    //////////////////////////////////////////////////////////////////////////////
    // LISTADO ///////////////////////////////////////////////////////////////////
    //////////////////////////////////////////////////////////////////////////////

    case 1:
      System.out.println("\nLISTADO");
      System.out.println("=======");

      for(i = 0; i < N; i++) {
```

```
        if (!articulo[i].getCodigo().equals("LIBRE")) {
          System.out.println(articulo[i]);
        }
      }

      break;

    ///////////////////////////////////////////////////////////////////////////
    // ALTA ///////////////////////////////////////////////////////////////////
    ///////////////////////////////////////////////////////////////////////////

    case 2:
      System.out.println("\nNUEVO ARTÍCULO");
      System.out.println("==============");

      // Busca la primera posición libre del array
      primeraLibre = 0;
      codigo = articulo[primeraLibre].getCodigo();
      while ((primeraLibre < N) && (!(codigo.equals("LIBRE")))) {
        primeraLibre++;
        if (primeraLibre < N) {
          codigo = articulo[primeraLibre].getCodigo();
        }
      }

      if (primeraLibre == N) {
        System.out.println("Lo siento, a base de datos está llena.");
      } else {

        // Introducción de datos

        System.out.println("Por favor, introduzca los datos del artículo.");
        System.out.print("Código: ");

        // Comprueba que el código introducido no se repita
        existeCodigo = true;
        while (existeCodigo) {
          existeCodigo = false;
          codigoIntroducido = System.console().readLine();

          for (i = 0; i < N; i++) {
            if (codigoIntroducido.equals(articulo[i].getCodigo())) {
              existeCodigo = true;
            }
          }
```

```
        if (existeCodigo) {
          System.out.println("Ese código ya existe en la base de datos.");
          System.out.print("Introduzca otro código: ");
        }
      } // while (existeCodigo)

      articulo[primeraLibre].setCodigo(codigoIntroducido);

      System.out.print("Descripcion: ");
      descripcionIntroducida = System.console().readLine();
      articulo[primeraLibre].setDescripcion(descripcionIntroducida);

      System.out.print("Precio de compra: ");
      precioDeCompraIntroducido = Double.parseDouble(System.console().readLine());
      articulo[primeraLibre].setPrecioDeCompra(precioDeCompraIntroducido);

      System.out.print("Precio de venta: ");
      precioDeVentaIntroducido = Double.parseDouble(System.console().readLine());
      articulo[primeraLibre].setPrecioDeVenta(precioDeVentaIntroducido);

      System.out.print("Stock: ");
      stockIntroducido = Integer.parseInt(System.console().readLine());
      articulo[primeraLibre].setStock(stockIntroducido);
    }

    break;

  //////////////////////////////////////////////////////////////////////////////
  // BAJA ////////////////////////////////////////////////////////////////////
  //////////////////////////////////////////////////////////////////////////////

  case 3:
    System.out.println("\nBAJA");
    System.out.println("====");

    System.out.print("Por favor, introduzca el código del artículo que desea dar de baja\
: ");

    codigoIntroducido = System.console().readLine();

    i = -1;
    codigo = "";
    do {
      i++;
      if (i < N) {
        codigo = articulo[i].getCodigo();
      }
```

```
    } while (!(codigo.equals(codigoIntroducido)) && (i < N));

    if (i == N) {
      System.out.println("Lo siento, el código introducido no existe.");
    } else {
      articulo[i].setCodigo("LIBRE");
      System.out.println("articulo borrado.");
    }

    break;

  //////////////////////////////////////////////////////////////////////////////
  // MODIFICACIÓN //////////////////////////////////////////////////////////////
  //////////////////////////////////////////////////////////////////////////////

  case 4:
    System.out.println("\nMODIFICACIÓN");
    System.out.println("============");

    System.out.print("Por favor, introduzca el código del artículo cuyos datos desea cam\
biar: ");
    codigoIntroducido = System.console().readLine();

    i = -1;
    do {
      i++;
    } while (!((articulo[i].getCodigo()).equals(codigoIntroducido)));

    System.out.println("Introduzca los nuevos datos del artículo o INTRO para dejarlos i\
gual.");

    System.out.println("Código: " + articulo[i].getCodigo());
    System.out.print("Nuevo código: ");
    codigoIntroducido = System.console().readLine();
    if (!codigoIntroducido.equals("")) {
      articulo[i].setCodigo(codigoIntroducido);
    }

    System.out.println("Descripción: " + articulo[i].getDescripcion());
    System.out.print("Nueva descripción: ");
    descripcionIntroducida = System.console().readLine();
    if (!descripcionIntroducida.equals("")) {
      articulo[i].setDescripcion(descripcionIntroducida);
    }

    System.out.println("Precio de compra: " + articulo[i].getPrecioDeCompra());
```

```
System.out.print("Nuevo precio de compra: ");
precioDeCompraIntroducidoString = System.console().readLine();
if (!precioDeCompraIntroducidoString.equals("")) {
  articulo[i].setPrecioDeCompra(Double.parseDouble(precioDeCompraIntroducidoString));
}
System.out.println("Precio de venta: " + articulo[i].getPrecioDeVenta());
System.out.print("Nuevo precio de venta: ");
precioDeVentaIntroducidoString = System.console().readLine();
if (!precioDeVentaIntroducidoString.equals("")) {
  articulo[i].setPrecioDeVenta(Double.parseDouble(precioDeVentaIntroducidoString));
}
System.out.println("Stock: " + articulo[i].getStock());
System.out.print("Nuevo stock: ");
stockIntroducidoString = System.console().readLine();
if (!stockIntroducidoString.equals("")) {
  articulo[i].setStock(Integer.parseInt(stockIntroducidoString));
}
break;


//////////////////////////////////////////////////////////////////////////////
// ENTRADA DE MERCANCÍA ///////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////

case 5:
  System.out.println("\nENTRADA DE MERCANCÍA");
  System.out.println("====================");

  System.out.print("Por favor, introduzca el código del artículo: ");
  codigoIntroducido = System.console().readLine();

  i = -1;
  codigo = "";
  do {
    i++;
    if (i < N) {
      codigo = articulo[i].getCodigo();
    }
  } while (!(codigo.equals(codigoIntroducido)) && (i < N));

  if (i == N) {
    System.out.println("Lo siento, el código introducido no existe.");
  } else {
    System.out.println("Entrada de mercancía del siguiente artículo: ");
    System.out.println(articulo[i]);
    System.out.print("Introduzca el número de unidades que entran al almacén: ");
```

```
              stockIntroducidoString = System.console().readLine();
              articulo[i].setStock(Integer.parseInt(stockIntroducidoString) + articulo[i].getS\
tock());

              System.out.println("La mercancía ha entrado en el almacén.");
            }

            break;


            ////////////////////////////////////////////////////////////////////////////////
            // SALIDA DE MERCANCÍA /////////////////////////////////////////////////////////
            ////////////////////////////////////////////////////////////////////////////////

            case 6:
              System.out.println("\nSALIDA DE MERCANCÍA");
              System.out.println("===================");

              System.out.print("Por favor, introduzca el código del artículo: ");
              codigoIntroducido = System.console().readLine();

              i = -1;
              codigo = "";
              do {
                i++;
                if (i < N) {
                  codigo = articulo[i].getCodigo();
                }
              } while (!(codigo.equals(codigoIntroducido)) && (i < N));

              if (i == N) {
                System.out.println("Lo siento, el código introducido no existe.");
              } else {
                System.out.println("Salida de mercancía del siguiente artículo: ");
                System.out.println(articulo[i]);
                System.out.print("Introduzca el número de unidades que desea sacar del almacén\
: ");

                stockIntroducido = Integer.parseInt(System.console().readLine());
                if (articulo[i].getStock() - stockIntroducido > 0) {
                  articulo[i].setStock(articulo[i].getStock() - stockIntroducido);
                  System.out.println("La mercancía ha salido del almacén.");
                } else {
                  System.out.println("Lo siento, no se pueden sacar tantas unidades.");
                }
              }

              break;
```

```
      } // switch
    } while (opcion != 7);
  }
}
```

Colecciones y diccionarios

Ejercicio 1

Fichero: S10Ejercicio01.java

```java
/**
 * 1. Crea un ArrayList con los nombres de 6 compañeros de clase. A
 *    continuación, muestra esos nombres por pantalla. Utiliza para
 *    ello un bucle for que recorra todo el ArrayList sin usar ningún
 *    índice.
 *
 * @author Luis José Sánchez
 */
import java.util.ArrayList;

public class S10Ejercicio01 {
  public static void main(String[] args) {

    ArrayList<String> a = new ArrayList<String>();

    a.add("José Manuel");
    a.add("Salvador");
    a.add("Rubén");
    a.add("Irene");
    a.add("Noemí");
    a.add("Begoña");

    System.out.println("Contenido de la lista: ");

    for(String nombre : a) {
      System.out.println(nombre);
    }
  }
}
```

Ejercicio 2

Fichero: S10Ejercicio02.java

```java
/**
 * 2. Realiza un programa que introduzca valores aleatorios (entre 0 y
 *    100) en un ArrayList y que luego calcule la suma, la media, el
 *    máximo y el mínimo de esos números. El tamaño de la lista también
 *    será aleatorio y podrá oscilar entre 10 y 20 elementos ambos
 *    inclusive.
 *
 * @author Luis José Sánchez
 */
import java.util.ArrayList;

public class S10Ejercicio02 {
  public static void main(String[] args) {

    int suma = 0;
    int minimo = 100;
    int maximo = 0;
    int tamano = (int)(Math.random() * 11 + 10);

    ArrayList<Integer> a = new ArrayList<Integer>();

    for (int i = 0; i < tamano; i++) {
      a.add((int)(Math.random()*101));
    }

    System.out.println("Lista generada: " + a);

    for(int n : a) {
      suma += n;

      if(n < minimo) {
        minimo = n;
      }

      if(n > maximo) {
        maximo = n;
      }
     }

    System.out.println("La suma total es : " + suma);
    System.out.println("La media es : " + suma / tamano);
    System.out.println("El mínimo es : " + minimo);
    System.out.println("El máximo es : " + maximo);
  }
}
```

Ejercicio 3

Fichero: S10Ejercicio03.java

```
/**
 * 3. Escribe un programa que ordene 10 números enteros introducidos por
 *    teclado y almacenados en un objeto de la clase ArrayList .
 *
 * @author Luis José Sánchez
 */
import java.util.ArrayList;
import java.util.Collections;

public class S10Ejercicio03 {
  public static void main(String[] args) {

    ArrayList<Integer> a = new ArrayList<Integer>();

    System.out.println("Introduzca 10 números enteros: ");

    for (int i = 0; i < 10; i++) {
      a.add(Integer.parseInt(System.console().readLine()));
    }

    System.out.println("Lista original: " + a);

    Collections.sort(a);

    System.out.println("Lista ordenada de menor a mayor: " + a);

    // Descomenta el siguiente bloque para ordenar la lista de mayor a
    // menor
    /*
      ArrayList<Integer> b = new ArrayList<Integer>();

      for(int n : a) {
        b.add(0, n);
      }

      System.out.print("Lista ordenada de mayor a menor: " + b);
    */
  }
}
```

Ejercicio 4

Fichero: S10Ejercicio04.java

```
/**
 * 4. Realiza un programa equivalente al anterior pero en esta ocasión,
 *    el programa debe ordenar palabras en lugar de números.
 *
 * @author Luis José Sánchez
 */
import java.util.ArrayList;
import java.util.Collections;

public class S10Ejercicio04 {
  public static void main(String[] args) {

  ArrayList<String> a = new ArrayList<String>();

  System.out.println("Introduzca 10 palabras: ");
  for (int i = 0; i < 10; i++) {
    a.add(System.console().readLine());
  }

  System.out.println("\nLista original:\n" + a);

  Collections.sort(a);

  System.out.println("\nLista ordenada alfabéticamente:\n" + a);
  }
}
```

Ejercicio 5

Fichero: ColeccionDeDiscosConArrayList.java

```
/**
 * 5. Realiza de nuevo el ejercicio de la colección de discos pero
 *    utilizando esta vez una lista para almacenar la información
 *    sobre los discos en lugar de un array convencional. Comprobarás
 *    que el código se simplifica notablemente ¿Cuánto ocupa el programa
 *    original hecho con un array? ¿Cuánto ocupa este nuevo programa
 *    hecho con una lista?
 *
 * @author Luis José Sánchez
 */
```

```
import java.util.ArrayList;
import java.util.Scanner;

public class ColeccionDeDiscosConArrayList {
  public static void main(String[] args) {

    ArrayList<Disco> album = new ArrayList<Disco>();

    int i;
    String codigoIntroducido;
    String autorIntroducido;
    String tituloIntroducido;
    String generoIntroducido;
    String duracionIntroducidaString;
    int opcion;
    int opcionListado;
    int duracionIntroducida;
    int limiteInferior;
    int limiteSuperior;

    Scanner s = new Scanner(System.in);

    // DISCOS PREDEFINIDOS
    album.add(new Disco("GASA41", "Wim Mertens", "Maximazing the Audience", "instrumental", 50\
));
    album.add(new Disco("FGHQ64", "Metallica", "Black album", "hard rock", 46));
    album.add(new Disco("TYUI89", "Supersubmarina", "Viento de cara", "pop rock", 42));

    // MENÚ
    do {
      System.out.println("\n\nCOLECCIÓN DE DISCOS");
      System.out.println("===================");
      System.out.println("1. Listado");
      System.out.println("2. Nuevo disco");
      System.out.println("3. Modificar");
      System.out.println("4. Borrar");
      System.out.println("5. Salir");
      System.out.print("Introduzca una opción: ");
      opcion = Integer.parseInt(s.nextLine());

      switch (opcion) {

      // LISTADO /////////////////////////////////////////////////////////////////
      case 1:
        do {
          System.out.println("\nLISTADO\n=======");
```

```
System.out.println("1. Completo");
System.out.println("2. Por autor");
System.out.println("3. Por género");
System.out.println("4. En un rango de duración");
System.out.println("5. Menú principal");
System.out.print("Introduzca una opción: ");
opcionListado = Integer.parseInt(s.nextLine());

switch (opcionListado) {

  case 1: // Listado completo ///////////////

    for(Disco d : album) {
      System.out.println(d);
    }
    break;

  case 2: // Listado por autor ///////////////

    System.out.print("Introduzca el autor: ");
    autorIntroducido = s.nextLine();

    for(Disco d : album) {
      if(d.getAutor().equals(autorIntroducido)) {
        System.out.println(d);
      }
    }
    break;

  case 3: // Listado por género ///////////////

    System.out.print("Introduzca el género: ");
    generoIntroducido = s.nextLine();

    for(Disco d : album) {
      if(d.getGenero().equals(generoIntroducido)) {
        System.out.println(d);
      }
    }
    break;

  case 4: // Listado en un rango de duración ////

    System.out.println("Establezca el intervalo para la duración");
    System.out.print("Introduzca el límite inferior de duración en minutos: ");
    limiteInferior = Integer.parseInt(s.nextLine());
```

```
              System.out.print("Introduzca el límite superior de duración en minutos: ");
              limiteSuperior = Integer.parseInt(s.nextLine());

              for(Disco d : album) {
                if((d.getDuracion() <= limiteSuperior) && (d.getDuracion() >= limiteInferior))\
 {
                  System.out.println(d);
                }
              }
          } // switch (opcionListado)

        } while (opcionListado != 5);

        break;

      // NUEVO DISCO //////////////////////////////////////////////////////////////
      case 2:
        System.out.println("\nNUEVO DISCO\n===========");

        System.out.println("Por favor, introduzca los datos del disco.");
        System.out.print("Código: ");
        codigoIntroducido = s.nextLine();

        while (album.contains(new Disco(codigoIntroducido,"", "", "", 0))) {
          System.out.println("Ese código ya existe en la base de datos.");
          System.out.print("Introduzca otro código: ");
          codigoIntroducido = s.nextLine();
        }

        System.out.print("Autor: ");
        autorIntroducido = s.nextLine();
        System.out.print("Título: ");
        tituloIntroducido = s.nextLine();
        System.out.print("Género: ");
        generoIntroducido = s.nextLine();
        System.out.print("Duración: ");
        duracionIntroducida = Integer.parseInt(s.nextLine());

        album.add(new Disco(codigoIntroducido, autorIntroducido, tituloIntroducido, generoIntr\
oducido, duracionIntroducida));

        break;

      // MODIFICAR ////////////////////////////////////////////////////////////////
      case 3:
        System.out.println("\nMODIFICAR\n===========");
```

```
        System.out.print("Por favor, introduzca el código del disco cuyos datos desea cambiar:\
 ");
        codigoIntroducido = s.nextLine();

        while (!album.contains(new Disco(codigoIntroducido,"", "", "", 0))) {
          System.out.print("Ese código no existe.\nIntroduzca otro código: ");
          codigoIntroducido = s.nextLine();
        }
        i = album.indexOf(new Disco(codigoIntroducido,"", "", "", 0));

        System.out.println("Introduzca los nuevos datos del disco o INTRO para dejarlos igual.\
");

        System.out.println("Código: " + album.get(i).getCodigo());
        System.out.print("Nuevo código: ");
        codigoIntroducido = s.nextLine();
        if (!codigoIntroducido.equals("")) {
          album.get(i).setCodigo(codigoIntroducido);
        }

        System.out.println("Autor: " + album.get(i).getAutor());
        System.out.print("Nuevo autor: ");
        autorIntroducido = s.nextLine();
        if (!autorIntroducido.equals("")) {
          album.get(i).setAutor(autorIntroducido);
        }

        System.out.println("Título: " + album.get(i).getTitulo());
        System.out.print("Nuevo título: ");
        tituloIntroducido = s.nextLine();
        if (!tituloIntroducido.equals("")) {
          album.get(i).setTitulo(tituloIntroducido);
        }

        System.out.println("Género: " + album.get(i).getGenero());
        System.out.print("Nuevo género: ");
        generoIntroducido = s.nextLine();
        if (!generoIntroducido.equals("")) {
          album.get(i).setGenero(generoIntroducido);
        }

        System.out.println("Duración: " + album.get(i).getDuracion());
        System.out.print("Duración: ");
        duracionIntroducidaString = s.nextLine();
        if (!duracionIntroducidaString.equals("")) {
```

```
          album.get(i).setDuracion(Integer.parseInt(duracionIntroducidaString));
        }

        break;

      // BORRAR //////////////////////////////////////////////////////////////////
      case 4:
        System.out.println("\nBORRAR\n======");

        System.out.print("Por favor, introduzca el código del disco que desea borrar: ");
        codigoIntroducido = s.nextLine();

        if (!album.contains(new Disco(codigoIntroducido,"", "", "", 0))) {
          System.out.println("Lo siento, el código introducido no existe.");
        } else {
          album.remove(album.indexOf(new Disco(codigoIntroducido,"", "", "", 0)));
          System.out.println("Album borrado.");
        }
      } // switch
    } while (opcion != 5);
  }
}
```

Fichero: Disco.java

```
public class Disco {
  private String codigo;
  private String autor;
  private String titulo;
  private String genero;
  private int duracion; // duración total en minutos

  public Disco(String c, String a, String t, String g, int d) {
    this.codigo = c;
    this.autor = a;
    this.titulo = t;
    this.genero = g;
    this.duracion = d;
  }

  public String getCodigo() {
    return codigo;
  }

  public void setCodigo(String codigo) {
    this.codigo = codigo;
```

```
  }

  public String getAutor() {
    return autor;
  }

  public void setAutor(String autor) {
    this.autor = autor;
  }

  public String getGenero() {
    return genero;
  }

  public void setGenero(String genero) {
    this.genero = genero;
  }

  public String getTitulo() {
    return titulo;
  }

  public void setTitulo(String titulo) {
    this.titulo = titulo;
  }

  public int getDuracion() {
    return duracion;
  }

  public void setDuracion(int duracion) {
    this.duracion = duracion;
  }

  @Override
  public String toString() {
    String cadena = "\n----------------------------------------";
    cadena += "\nCódigo: " + this.codigo;
    cadena += "\nAutor: " + this.autor;
    cadena += "\nTítulo: " + this.titulo;
    cadena += "\nGénero: " + this.genero;
    cadena += "\nDuración: " + this.duracion;
    cadena += "\n----------------------------------------";

    return cadena;
  }
```

```
  // Considero que dos discos son iguales si tienen el mismo código.
  // Es obligatorio pasar un objeto genérico como parámetro.
  @Override
  public boolean equals(Object d) {
    return (this.codigo).equals(((Disco)d).getCodigo());
  }
}
```

Ejercicio 6

Fichero: S10Ejercicio06.java

```
/**
 * 6. Implementa el control de acceso al área restringida de un
 *    programa. Se debe pedir un nombre de usuario y una contraseña. Si
 *    el usuario introduce los datos correctamente, el programa dirá
 *    "Ha accedido al área restringida". El usuario tendrá un máximo de
 *    3 oportunidades. Si se agotan las oportunidades el programa dirá
 *    "Lo siento, no tiene acceso al área restringida". Los nombres de
 *    usuario con sus correspondientes contraseñas deben estar
 *    almacenados en una estructura de la clase HashMap.
 *
 * @author Luis José Sánchez
 */
import java.util.*;

public class S10Ejercicio06 {
  public static void main(String[] args) {

    HashMap<String, String> m = new HashMap<String, String>();

    String usuario;
    String clave;
    boolean entra = false;
    int oportunidades = 3;

    m.put("admin", "224477");
    m.put("maria", "ztf99");
    m.put("pepe", "zxcvb");

    System.out.println("Por favor, introduzca nombre de usuario y "
          + "contraseña para acceder al área restringida.\nDispone de "
          + "3 intentos.");
```

```
    do {
      System.out.print("Usuario: ");
      usuario = System.console().readLine();
      System.out.print("Contraseña: ");
      clave = System.console().readLine();

      if (m.containsKey(usuario)) { // el usuario existe
        if (m.get(usuario).equals(clave)) {
          System.out.println("Ha accedido al área restringida");
          entra = true;
        } else {
          System.out.println("Contraseña incorrecta");
        }
      } else { // el usuario no existe
        System.out.println("El usuario introducido no existe");
      }

      oportunidades--;

      if (!entra && (oportunidades > 0)) {
        System.out.println("Le quedan " + (oportunidades) + " oportunidades");
      }

    } while ((!entra) && (oportunidades > 0));

    if (!entra){
      System.out.print("Lo siento, no tiene acceso al área restringida");
    }
  }
}
```

Ejercicio 7

Fichero: Eurocoin.java

```
/**
 * 7. La máquina Eurocoin genera una moneda de curso legal cada vez que
 *    se pulsa un botón siguiendo la siguiente pauta: o bien coincide el
 *    valor con la moneda anteriormente generada - 1 céntimo, 2 céntimos,
 *    5 céntimos, 10 céntimos, 25 céntimos, 50 céntimos, 1 euro o 2 euros -
 *    o bien coincide la posición – cara o cruz. Simula, mediante un
 *    programa, la generación de 6 monedas aleatorias siguiendo la pauta
 *    correcta. Cada moneda generada debe ser una instancia de la clase
 *    Moneda y la secuencia se debe ir almacenando en una lista.
 *    Ejemplo:
```

```
 *    2 céntimos - cara
 *    2 céntimos - cruz
 *    50 céntimos - cruz
 *    1 euro - cruz
 *    1 euro - cara
 *    10 céntimos - cara
 *
 *    @author Luis José Sánchez
 */
import java.util.ArrayList;

public class Eurocoin {

  public static void main(String[] args) {
    ArrayList<Moneda> m = new ArrayList<Moneda>();

    Moneda monedaAux = new Moneda();
    m.add(monedaAux);

    String ultimaPosicion = monedaAux.getPosicion();
    String ultimaCantidad = monedaAux.getCantidad();

    for (int i = 0; i < 5; i++) {
      do {
        monedaAux = new Moneda();
      } while (!((monedaAux.getPosicion()).equals(ultimaPosicion)) && !((monedaAux.getCantidad\
()).equals(ultimaCantidad)));

      m.add(monedaAux);
      ultimaPosicion = monedaAux.getPosicion();
      ultimaCantidad = monedaAux.getCantidad();
    }

    for (Moneda mo : m) {
      System.out.println(mo);
    }
  }
}
```

Fichero: Moneda.java

```
public class Moneda {
  private static String posiciones[] = {"cara", "cruz"};
  private static String cantidades[] = {"1 céntimo", "2 céntimos", "5 céntimos", "10 céntimos"\
, "25 céntimos", "50 céntimos", "1 euro", "2 euros"};
  private String cantidad;
  private String posicion;

  public Moneda() {
    this.posicion = posiciones[(int)(Math.random()*2)];
    this.cantidad = cantidades[(int)(Math.random()*8)];
  }

  public String getPosicion() {
    return this.posicion;
  }

  public String getCantidad() {
    return this.cantidad;
  }

  public String toString() {
    return this.cantidad + " - " + this.posicion;
  }
}
```

Ejercicio 8

Fichero: Ejercicio08.java

```
/**
 * 8. Realiza un programa que escoja al azar 10 cartas de la baraja
 *    española (10 objetos de la clase <code>Carta</code>). Emplea un
 *    objeto de la clase <code>ArrayList</code> para almacenarlas y
 *    asegúrate de que no se repite ninguna.
 *
 *    @author Luis José Sánchez
 */
import java.util.ArrayList;

public class Ejercicio08 {
  public static void main(String[] args) {

    ArrayList<Carta> c = new ArrayList<Carta>();

    Carta cartaAux = new Carta();
```

```
    c.add(cartaAux);

    for (int i = 0; i < 9; i++) {
      do {
        cartaAux = new Carta();
      } while (c.contains(cartaAux));

      c.add(cartaAux);
    }

    for (Carta miCarta: c) {
      System.out.println(miCarta);
    }
  }
}
```

Fichero: Carta.java

```
import java.util.Objects;

public class Carta {

  private static String[] n = {"as", "dos", "tres", "cuatro", "cinco", "seis", "siete", "sota"\
, "caballo", "rey"};
  private static String[] p = {"bastos", "copas", "espadas", "oros"};

  private String numero;
  private String palo;

  public Carta() {
    this.numero = n[(int)(Math.random()*10)];
    this.palo = p[(int)(Math.random()*4)];
  }

  public String getNumero() {
    return numero;
  }

  public String getPalo() {
    return palo;
  }

  @Override
  public String toString() {
    return this.numero + " de " + this.palo;
  }
```

```
  @Override
  public boolean equals(Object obj) {
    if (obj == null) {
      return false;
    }
    if (getClass() != obj.getClass()) {
      return false;
    }
    final Carta other = (Carta) obj;
    if (!Objects.equals(this.numero, other.numero)) {
      return false;
    }
    if (!Objects.equals(this.palo, other.palo)) {
      return false;
    }
    return true;
  }
}
```

Ejercicio 9

Fichero: Ejercicio09.java

```
/*
 * 9. Modifica el programa anterior de tal forma que las cartas se
 *    muestren ordenadas. Primero se ordenarán por palo: bastos,
 *    copas, espadas, oros. Cuando coincida el palo, se ordenará por
 *    número: as, 2, 3, 4, 5, 6, 7, sota, caballo, rey.
 *
 *    @author Luis José Sánchez
 */
import java.util.ArrayList;
import java.util.Collections;

public class Ejercicio09 {

  public static void main(String[] args) {

    ArrayList<Carta> c = new ArrayList<Carta>();

    Carta cartaAux = new Carta();
    c.add(cartaAux);

    for (int i = 0; i < 9; i++) {
```

```
      do {
        cartaAux = new Carta();
      } while (c.contains(cartaAux));

      c.add(cartaAux);
    }

    Collections.sort(c);

    for (Carta miCarta: c) {
      System.out.println(miCarta);
    }
  }
}
```

Fichero: Carta.java

```
import java.util.Objects;

public class Carta implements Comparable<Carta>{

  private static String[] n = {"as", "dos", "tres", "cuatro", "cinco", "seis", "siete", "sota"\
, "caballo", "rey"};
  private static String[] p = {"bastos", "copas", "espadas", "oros"};

  private Integer numero;
  private String palo;

  public Carta() {
    this.numero = (int)(Math.random()*10);
    this.palo = p[(int)(Math.random()*4)];
  }

  public int getNumero() {
    return numero;
  }

  public String getPalo() {
    return palo;
  }

  @Override
  public String toString() {
    return n[numero] + " de " + palo;
  }
```

```
  @Override
  public boolean equals(Object obj) {
    if (obj == null) {
      return false;
    }
    if (getClass() != obj.getClass()) {
      return false;
    }
    final Carta other = (Carta) obj;
    if (!Objects.equals(this.numero, other.numero)) {
      return false;
    }
    if (!Objects.equals(this.palo, other.palo)) {
      return false;
    }
    return true;
  }

  @Override
  public int compareTo(Carta c) {
    if (palo.equals(c.getPalo())) {
      return numero.compareTo(c.getNumero());
    } else {
      return palo.compareTo(c.getPalo());
    }
  }
}
```

Ejercicio 10

Fichero: S10Ejercicio10.java

```
/**
 * 10. Crea un mini-diccionario español-inglés que contenga, al menos,
 *     20 palabras (con su traducción).
 *     <p>
 *     Utiliza un objeto de la clase <code>HashMap</code> para almacenar
 *     las parejas de palabras. El programa pedirá una palabra en
 *     español y dará la correspondiente traducción en inglés.
 */
import java.util.HashMap;

public class S10Ejercicio10 {
  public static void main(String[] args) {
```

```
    HashMap<String, String> m = new HashMap<String, String>();

    m.put("ordenador", "computer");
    m.put("gato", "cat");
    m.put("rojo", "red");
    m.put("árbol", "tree");
    m.put("pingüino", "penguin");
    m.put("sol", "sun");
    m.put("agua", "water");
    m.put("viento", "wind");
    m.put("siesta", "siesta");
    m.put("arriba", "up");
    m.put("ratón", "mouse");
    m.put("estadio", "arena");
    m.put("calumnia", "aspersion");
    m.put("aguacate", "avocado");
    m.put("cuerpo", "body");
    m.put("concurso", "contest");
    m.put("cena", "dinner");
    m.put("salida", "exit");
    m.put("lenteja", "lentil");
    m.put("cacerola", "pan");
    m.put("pastel", "pie");
    m.put("membrillo", "quince");

    System.out.print("Introduzca una palabra en español: ");
    String palabraIntro = System.console().readLine();

    if (m.containsKey(palabraIntro)) {
      System.out.println(palabraIntro + " en inglés es " + m.get(palabraIntro));
    } else {
      System.out.print("Lo siento, no conozco esa palabra.");
    }
  }
}
```

Ejercicio 11

Fichero: S10Ejercicio11.java

```java
/**
 * 11. Realiza un programa que escoja al azar 5 palabras en español del
 *     minidiccionario del ejercicio anterior. El programa irá pidiendo
 *     que el usuario teclee la traducción al inglés de cada una de las
 *     palabras y comprobará si son correctas. Al final, el programa
 *     deberá mostrar cuántas respuestas son válidas y cuántas erróneas.
 *
 *    @author Luis José Sánchez
 */
import java.util.HashMap;
import java.util.ArrayList;

public class S10Ejercicio11 {
  public static void main(String[] args) {

    HashMap<String, String> m = new HashMap<String, String>();

    m.put("ordenador", "computer");
    m.put("gato", "cat");
    m.put("rojo", "red");
    m.put("árbol", "tree");
    m.put("pingüino", "penguin");
    m.put("sol", "sun");
    m.put("agua", "water");
    m.put("viento", "wind");
    m.put("siesta", "siesta");
    m.put("arriba", "up");
    m.put("ratón", "mouse");
    m.put("estadio", "arena");
    m.put("calumnia", "aspersion");
    m.put("aguacate", "avocado");
    m.put("cuerpo", "body");
    m.put("concurso", "contest");
    m.put("cena", "dinner");
    m.put("salida", "exit");
    m.put("lenteja", "lentil");
    m.put("cacerola", "pan");
    m.put("pastel", "pie");
    m.put("membrillo", "quince");

    int i = 0;
    int numero;

    // guarda las claves en un Array de String
    String[] a = m.keySet().toArray(new String[0]);
```

```
    // genera 5 números aleatorios que no se repiten
    ArrayList<Integer> n = new ArrayList<Integer>();
    n.add((int)(Math.random()*22));

    for (i = 0; i < 4; i++) {
      do {
        numero = (int)(Math.random()*22);
      } while (n.contains(numero));
      n.add(numero);
    }

    // muestra las palabras en español y pregunta por su traducción
    int puntos = 0;
    System.out.println("Mostraré la palabra en español y usted tendrá que traducirla al inglés\
.");
    for (i = 0; i < 5; i++) {
      System.out.print(a[n.get(i)] + ": ");
      String palabraIntro = System.console().readLine();
      if (palabraIntro.equals(m.get(a[n.get(i)]))) {
        System.out.println("¡Correcto!");
        puntos++;
      } else {
        System.out.println("Respuesta incorrecta :(");
        System.out.println("La respuesta correcta es " + m.get(a[n.get(i)]));
      }
    }

    System.out.println("Ha obtenido " + puntos + " puntos.");
  }
}
```

Ejercicio 12

Fichero: Ejercicio12.java

```
/**
 * 12. Escribe un programa que genere una secuencia de 5 cartas de la
 *     baraja española y que sume los puntos según el juego de la
 *     brisca. El valor de las cartas se debe guardar en una estructura
 *     <code>HashMap</code> que debe contener parejas (figura, valor),
 *     por ejemplo ("caballo", 3).
 *     <p>
 *     La secuencia de cartas debe ser una estructura de la clase <code>
 *     ArrayList</code> que contiene objetos de la clase <code>Carta
 *     </code>. El valor de las cartas es el siguiente: as □ 11, tres □
```

```
 *     10, sota □ 2, caballo □ 3, rey □ 4; el resto de cartas no vale
 *     nada.
 *     <p>
 *     <code>
 *     <pre>
 *     Ejemplo:
 *     as de oros
 *     cinco de bastos
 *     caballo de espadas
 *     sota de copas
 *     tres de oros
 *     Tienes 26 puntos
 *     </pre>
 *     </code>
 *    @author Luis José Sánchez
 */
import java.util.ArrayList;
import java.util.HashMap;

public class Ejercicio12 {
  public static void main(String[] args) {

    ArrayList<Carta> c = new ArrayList<Carta>();
    HashMap<String, Integer> h = new HashMap<String, Integer>();
    int puntos = 0;

    h.put("as", 11);
    h.put("dos", 0);
    h.put("tres", 10);
    h.put("cuatro", 0);
    h.put("cinco", 0);
    h.put("seis", 0);
    h.put("siete", 0);
    h.put("sota", 2);
    h.put("caballo", 3);
    h.put("rey", 4);

    Carta cartaAux = new Carta();
    c.add(cartaAux);

    for (int i = 0; i < 5; i++) {
      do {
        cartaAux = new Carta();
      } while (c.contains(cartaAux));

      c.add(cartaAux);
```

```
    }


    for (Carta miCarta: c) {
      System.out.println(miCarta);
      puntos += h.get(miCarta.getFigura());
    }

    System.out.println("Tienes " + puntos + " puntos.");
  }
}
```

Fichero: Carta.java

```
import java.util.Objects;

public class Carta implements Comparable<Carta>{

  private static String[] n = {"as", "dos", "tres", "cuatro", "cinco", "seis", "siete", "sota"\
, "caballo", "rey"};
  private static String[] p = {"bastos", "copas", "espadas", "oros"};

  private Integer numero;
  private String palo;

  public Carta() {
    this.numero = (int)(Math.random()*10);
    this.palo = p[(int)(Math.random()*4)];
  }

  public int getNumero() {
    return numero;
  }

  public String getFigura() {
    return n[numero];
  }

  public String getPalo() {
    return palo;
  }

  @Override
  public String toString() {
    return n[numero] + " de " + palo;
  }
```

```
  @Override
  public boolean equals(Object obj) {
    if (obj == null) {
      return false;
    }
    if (getClass() != obj.getClass()) {
      return false;
    }
    final Carta other = (Carta) obj;
    if (!Objects.equals(this.numero, other.numero)) {
      return false;
    }
    if (!Objects.equals(this.palo, other.palo)) {
      return false;
    }
    return true;
  }

  @Override
  public int compareTo(Carta c) {
    if (palo.equals(c.getPalo())) {
      return numero.compareTo(c.getNumero());
    } else {
      return palo.compareTo(c.getPalo());
    }
  }
}
```

Ejercicio 13

Fichero: Articulo.java

```
public class Articulo {
  private String codigo;
  private String descripcion;
  private double precioDeCompra;
  private double precioDeVenta;
  private int stock;

  public Articulo(String co, String de, double pc, double pv, int st) {
    this.codigo = co;
    this.descripcion = de;
    this.precioDeCompra = pc;
    this.precioDeVenta = pv;
```

```
    this.stock = st;
  }

  public Articulo(String co) {
    this.codigo = co;
  }

  public String getCodigo() {
    return codigo;
  }

  public void setCodigo(String codigo) {
    this.codigo = codigo;
  }

  public String getDescripcion() {
    return descripcion;
  }

  public void setDescripcion(String descripcion) {
    this.descripcion = descripcion;
  }

  public double getPrecioDeCompra() {
    return precioDeCompra;
  }

  public void setPrecioDeCompra(double precioDeCompra) {
    this.precioDeCompra = precioDeCompra;
  }

  public double getPrecioDeVenta() {
    return precioDeVenta;
  }

  public void setPrecioDeVenta(double precioDeVenta) {
    this.precioDeVenta = precioDeVenta;
  }

  public int getStock() {
    return stock;
  }

  public void setStock(int stock) {
    this.stock = stock;
  }
```

```
  @Override
  public String toString() {
    String cadena = "-------------------------------------------";
    cadena += "\nCódigo: " + this.codigo;
    cadena += "\nDescripción: " + this.descripcion;
    cadena += "\nPrecio de compra: " + this.precioDeCompra;
    cadena += "\nPrecio de venta: " + this.precioDeVenta;
    cadena += "\nStock: " + this.stock + " unidades";
    cadena += "\n-----------------------------------------";
    return cadena;
  }
}
```

Fichero: GestisimalMejorado.java

```
/**
 * 13. Modifica el programa Gestisimal realizado anteriormente añadiendo
 *     las siguientes mejoras:
 *     <ul>
 *      <li>Utiliza una lista en lugar de un array para el
 *          almacenamiento de los datos.
 *      <li>Comprueba la existencia del código en el alta, la baja y la
 *          modificación de artículos para evitar errores.
 *      <li>Cambia la opción "Salida de stock" por "Venta". Esta nueva
 *          opción permitirá hacer una venta de varios artículos y
 *          emitir la factura correspondiente. Se debe preguntar por los
 *          códigos y las cantidades de cada artículo que se quiere
 *          comprar. Aplica un 21% de IVA.
 *      </ul>
 *
 *  @author Luis José Sánchez
 */
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;

public class GestisimalMejorado {

  static ArrayList<Articulo> a = new ArrayList<Articulo>();

  public static void main(String[] args) {

    HashMap<String, Integer> lineasFra = new HashMap<String, Integer>();

    int opcion;
```

```
int opcion2;
int i;
int stockIntro;
int unidades = 0;
int unidadesEnFactura = 0;
double precioDeCompraIntro;
double precioDeVentaIntro;
double subtotal;
double baseImponible;
double totalFactura;
String codigo;
String codigoIntro = "";
String descripcionIntro;
String precioDeCompraIntroString;
String precioDeVentaIntroString;
String stockIntroString;

// Menu ///////////////////////////////////////////////////////////////
do {
  System.out.println("\n\nG E S T I S I M A L");
  System.out.println("===================");
  System.out.println("1. Listado");
  System.out.println("2. Alta");
  System.out.println("3. Baja");
  System.out.println("4. Modificación");
  System.out.println("5. Entrada de mercancía");
  System.out.println("6. Venta");
  System.out.println("7. Salir");
  System.out.print("Introduzca una opción: ");
  opcion = Integer.parseInt(System.console().readLine());

  switch (opcion) {

    case 1: // Listado ///////////////////////////////////////////////
      System.out.println("\nLISTADO\n=======");

      for(Articulo aux : a) {
        System.out.println(aux);
      }
      break;

    case 2: // Alta //////////////////////////////////////////////////
      System.out.println("\nNUEVO ARTÍCULO\n==============");

      System.out.println("Por favor, introduzca los datos del artículo.");
      System.out.print("Código: ");
```

```
          do {
            codigoIntro = System.console().readLine();
            if (posicion(codigoIntro) != -1) {
              System.out.print("Ese código ya existe.\nIntroduzca otro código: ");
            }
          } while  (posicion(codigoIntro) != -1);

          System.out.print("Descripcion: ");
          descripcionIntro = System.console().readLine();

          System.out.print("Precio de compra: ");
          precioDeCompraIntro = Double.parseDouble(System.console().readLine());

          System.out.print("Precio de venta: ");
          precioDeVentaIntro = Double.parseDouble(System.console().readLine());

          System.out.print("Stock: ");
          stockIntro = Integer.parseInt(System.console().readLine());

          a.add(new Articulo(codigoIntro, descripcionIntro, precioDeCompraIntro, precioDeVenta\
Intro, stockIntro));

          break;

        case 3: // Baja //////////////////////////////////////////////////
          System.out.println("\nBAJA\n====");
          System.out.print("Por favor, introduzca el código del artículo que desea dar de baja\
: ");
          codigoIntro = System.console().readLine();

          if (posicion(codigoIntro) == -1) {
            System.out.println("Lo siento, el código introducido no existe.");
          } else {
            a.remove(posicion(codigoIntro));
            System.out.println("articulo borrado.");
          }

          break;

        case 4: // Modificación ////////////////////////////////////////
          System.out.println("\nMODIFICACIÓN\n============");
          System.out.print("Por favor, introduzca el código del artículo cuyos datos desea cam\
biar: ");

          do {
```

```
          codigoIntro = System.console().readLine();
          if (posicion(codigoIntro) != -1) {
            System.out.println("No hay ningún artículo con ese código.\nIntroduzca otro códi\
go: ");
          }
        } while  (posicion(codigoIntro) == -1);

        i = posicion(codigoIntro);

        System.out.println("Introduzca los nuevos datos del artículo o INTRO para dejarlos i\
gual.");

        System.out.println("Código: " + a.get(i).getCodigo());
        System.out.print("Nuevo código: ");
        codigoIntro = System.console().readLine();
        if (!codigoIntro.equals("")) {
          a.get(i).setCodigo(codigoIntro);
        }

        System.out.println("Descripción: " + a.get(i).getDescripcion());
        System.out.print("Nueva descripción: ");
        descripcionIntro = System.console().readLine();
        if (!descripcionIntro.equals("")) {
          a.get(i).setDescripcion(descripcionIntro);
        }

        System.out.println("Precio de compra: " + a.get(i).getPrecioDeCompra());
        System.out.print("Nuevo precio de compra: ");
        precioDeCompraIntroString = System.console().readLine();
        if (!precioDeCompraIntroString.equals("")) {
          a.get(i).setPrecioDeCompra(Double.parseDouble(precioDeCompraIntroString));
        }

        System.out.println("Precio de venta: " + a.get(i).getPrecioDeVenta());
        System.out.print("Nuevo precio de venta: ");
        precioDeVentaIntroString = System.console().readLine();
        if (!precioDeVentaIntroString.equals("")) {
          a.get(i).setPrecioDeVenta(Double.parseDouble(precioDeVentaIntroString));
        }

        System.out.println("Stock: " + a.get(i).getStock());
        System.out.print("Nuevo stock: ");
        stockIntroString = System.console().readLine();
        if (!stockIntroString.equals("")) {
          a.get(i).setStock(Integer.parseInt(stockIntroString));
        }
```

```
          break;

          case 5: // Entrada de mercancía //////////////////////////////
            System.out.println("\nENTRADA DE MERCANCÍA\n===================");
            System.out.print("Por favor, introduzca el código del artículo: ");
            codigoIntro = System.console().readLine();

            do {
                codigoIntro = System.console().readLine();
                if (posicion(codigoIntro) != -1) {
                  System.out.println("No hay ningún artículo con ese código.\nIntroduzca otro \
código: ");
                }
            } while  (posicion(codigoIntro) == -1);

            i = posicion(codigoIntro);

            System.out.println("Entrada de mercancía del siguiente artículo: ");
            System.out.println(a.get(i));
            System.out.print("Introduzca el número de unidades que entran al almacén: ");
            stockIntro = Integer.parseInt(System.console().readLine());
            a.get(i).setStock(stockIntro + a.get(i).getStock());
            System.out.println("La mercancía ha entrado en el almacén.");

            break;

            case 6: // Venta
              System.out.println("\nVENTA\n=====");

              do {
                  System.out.println("\n1. Añadir artículo");
                  System.out.println("2. Generar factura");
                  System.out.println("3. Cancelar");
                  System.out.print("Introduzca una opción: ");
                  opcion2 = Integer.parseInt(System.console().readLine());

                  switch (opcion2) {

                    case 1: // Añadir línea ///////////////////////////
                      System.out.print("Por favor, introduzca el código del artículo: ");
                      codigoIntro = System.console().readLine();
                      i = posicion(codigoIntro);

                      if (i == -1) {
                        System.out.println("No hay ningún artículo con ese código.");
```

```
          } else {
            System.out.println(a.get(i));

            if (lineasFra.containsKey(codigoIntro)) {
              unidadesEnFactura = lineasFra.get(codigoIntro);
            } else {
              unidadesEnFactura = 0;
            }

            System.out.println("Unidades en la factura provisional: " + unidadesEn\
Factura);

            System.out.print("Unidades que quiere incorporar a la factura: ");
            unidades = Integer.parseInt(System.console().readLine());

            if ((a.get(i).getStock()) - unidadesEnFactura < unidades) {
              System.out.println("No hay suficiente stock. Puede añadir a la venta\
 un máximo de " + (a.get(i).getStock() - unidadesEnFactura) + " unidades de ese producto.");
            } else if (lineasFra.containsKey(codigoIntro)) {
              lineasFra.put(codigoIntro, lineasFra.get(codigoIntro) + unidades);
            } else {
              lineasFra.put(codigoIntro, unidades);
            }
          }

          // Muestra las líneas
          System.out.println("\n\n CÓDIGO |    DESCRIPCIÓN    | UNIDADES | PRECIO \
UNID. | SUBTOTAL");
          System.out.println("----------------------------------------------------\
--------------");
          for (Map.Entry pareja: lineasFra.entrySet()) {
            codigo = pareja.getKey().toString();
            i = posicion(codigo);
            unidades = Integer.parseInt(pareja.getValue().toString());
            subtotal = unidades * a.get(i).getPrecioDeVenta();
            System.out.printf(" %6s | %17s | %8d | %12.2f | %8.2f\n", codigo, a.ge\
t(i).getDescripcion(), unidades, a.get(i).getPrecioDeVenta(), subtotal);
          }

          break;

          case 2: // Genera la factura //////////////////////
            baseImponible = 0;
            System.out.println("\n\n CÓDIGO |   DESCRIPCIÓN   | UNIDADES | PRECIO \
UNID. | SUBTOTAL");
            System.out.println("--------------------------------------------------\
```

```
--------------");
                        for (Map.Entry pareja: lineasFra.entrySet()) {
                          codigo = pareja.getKey().toString();
                          i = posicion(codigo);
                          unidades = Integer.parseInt(pareja.getValue().toString());
                          subtotal = unidades * a.get(i).getPrecioDeVenta();
                          System.out.printf(" %6s | %15s | %8d | %12.2f | %8.2f\n", codigo, a.\
get(i).getDescripcion(), unidades, a.get(i).getPrecioDeVenta(), subtotal);
                          baseImponible += subtotal;
                          a.get(i).setStock(a.get(i).getStock() - unidades); // decrementa el \
stock
                        }

                        System.out.println("----------------------------------------------------\
--------------");
                        System.out.printf("                                      BASE IMPONIBL\
E: %8.2f \n", baseImponible);
                        System.out.printf("                                           IVA (21%\
%): %8.2f \n", baseImponible * 0.21);
                        System.out.printf("                                                TOTA\
L: %8.2f \n", baseImponible * 1.21);

                        System.out.println("\n\nFactura generada.\nPulse INTRO para volver al \
menú principal.");
                        System.console().readLine();

                        break;
            } // switch (venta)

          } while (opcion2 == 1);

          break;

      } // switch (menú principal)

    } while (opcion != 7);

  } // main

  /**
  /* Devuelve la posición de un artículo en el <code>ArrayList</code> o
   * <code>-1</code> si no existe.
   *
   * @param  codigo código del artículo que se buscará dentro del <code>
   *                ArrayList</code> que contiene todos los artículos
   * @return        posición que ocupa el artículo dentro del <code>
```

```
   *               ArrayList</code> o <code>-1</code> si no existe
   */
  static public int posicion(String codigo) {
    int i = -1;
    for (Articulo aux : a) {
      i++;
      if (aux.getCodigo().equals(codigo)) {
        return i;
      }
    }
    return -1;
  }
}
```

Ejercicio 14

Fichero: SupermercadoEcologico.java

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Scanner;

public class SupermercadoEcologico {

  public static void main(String[] args) {
    HashMap<String, Double> productos = new HashMap<String, Double>();
    productos.put("avena", 2.21);
    productos.put("garbanzos", 2.39);
    productos.put("tomate", 1.59);
    productos.put("jengibre", 3.13);
    productos.put("quinoa", 4.50);
    productos.put("guisantes", 1.60);

    Scanner s = new Scanner(System.in);
    String productoIntroducido = "";
    int cantidadIntroducida = 0;

    ArrayList<String> listaProductos = new ArrayList<>();
    ArrayList<Integer> listaCantidades = new ArrayList<>();

    do {
      System.out.print("Producto: ");
      productoIntroducido = s.nextLine();

      if (!productoIntroducido.equals("fin")) {
```

```
        System.out.print("Cantidad: ");
        cantidadIntroducida = Integer.parseInt(s.nextLine());
        listaProductos.add(productoIntroducido);
        listaCantidades.add(cantidadIntroducida);
      }

    } while (!productoIntroducido.equals("fin"));

    System.out.println("Producto Precio Cantidad Subtotal");
    System.out.println("---------------------------------");

    double total = 0;

    for (int i = 0; i < listaProductos.size(); i++) {
      String producto = listaProductos.get(i);
      double precio = productos.get(producto);
      int cantidad = listaCantidades.get(i);
      double subtotal = precio * cantidad;
      total += subtotal;
      System.out.printf("%-8s %7.2f %6d  %7.2f\n", producto, precio, cantidad, subtotal);
    }

    System.out.println("---------------------------------");
    System.out.printf("TOTAL: %.2f", total);
  }
}
```

Ejercicio 15

Fichero: SupermercadoEcologicoMejorado.java

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Scanner;

public class SupermercadoEcologicoMejorado {

  public static void main(String[] args) {
    HashMap<String, Double> productos = new HashMap<String, Double>();
    productos.put("avena", 2.21);
    productos.put("garbanzos", 2.39);
    productos.put("tomate", 1.59);
    productos.put("jengibre", 3.13);
    productos.put("quinoa", 4.50);
    productos.put("guisantes", 1.60);
```

```
Scanner s = new Scanner(System.in);
String productoIntroducido = "";
int cantidadIntroducida = 0;

ArrayList<String> listaProductos = new ArrayList<>();
ArrayList<Integer> listaCantidades = new ArrayList<>();

do {
  System.out.print("Producto: ");
  productoIntroducido = s.nextLine();

  if (!productoIntroducido.equals("fin")) {
    System.out.print("Cantidad: ");
    cantidadIntroducida = Integer.parseInt(s.nextLine());

    // Comprueba si ya existe el producto
    if (listaProductos.contains(productoIntroducido)) {
      int posicion = listaProductos.indexOf(productoIntroducido);
      listaCantidades.set(posicion, listaCantidades.get(posicion) + cantidadIntroducida);
    } else {
      listaProductos.add(productoIntroducido);
      listaCantidades.add(cantidadIntroducida);
    }
  }

} while (!productoIntroducido.equals("fin"));

System.out.print("Introduzca código de descuento (INTRO si no tiene ninguno): ");
String codigoDto = s.nextLine();

System.out.println("Producto Precio Cantidad Subtotal");
System.out.println("---------------------------------");

double total = 0;

for (int i = 0; i < listaProductos.size(); i++) {
  String producto = listaProductos.get(i);
  double precio = productos.get(producto);
  int cantidad = listaCantidades.get(i);
  double subtotal = precio * cantidad;
  total += subtotal;
  System.out.printf("%-8s %7.2f %6d  %7.2f\n", producto, precio, cantidad, subtotal);
}

double descuento = 0;
```

```
    if (codigoDto.equals("ECODTO")) {
      descuento = total / 10.0;
      total -= descuento;
    }

    System.out.println("---------------------------------");
    System.out.printf("Descuento: %.2f\n", descuento);
    System.out.println("---------------------------------");
    System.out.printf("TOTAL: %.2f\n", total);
  }
}
```

Ejercicio 16

Fichero: S10Ejercicio16.java

```
import java.util.HashMap;
import java.util.Scanner;

public class S10Ejercicio16 {

  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);

    HashMap<String, String> capitales = new HashMap<>();

    capitales.put("España", "Madrid");
    capitales.put("Portugal", "Lisboa");
    capitales.put("Francia", "París");

    String cadenaIntroducida = "";

    do {
      System.out.print("Escribe el nombre de un país y te diré su capital: ");
      cadenaIntroducida = s.nextLine();

      if (!cadenaIntroducida.equals("salir")) {
        if (capitales.containsKey(cadenaIntroducida)) {
          System.out.print("La capital de " + cadenaIntroducida);
          System.out.println(" es " + capitales.get(cadenaIntroducida));
        } else {
          System.out.print("No conozco la respuesta ");
          System.out.print("¿cuál es la capital de " + cadenaIntroducida + "?: ");
          String capital = s.nextLine();
          capitales.put(cadenaIntroducida, capital);
```

```
          System.out.println("Gracias por enseñarme nuevas capitales");
        }
      }
    } while (!cadenaIntroducida.equals("salir"));
  }

}
```

Ejercicio 17

Fichero: Elemento.java

```
public class Elemento {

  private String producto;
  private double precio;
  private int cantidad;

  public Elemento(String producto, double precio, int cantidad) {
    this.producto = producto;
    this.precio = precio;
    this.cantidad = cantidad;
  }

  public String getProducto() {
    return producto;
  }

  public void setProducto(String producto) {
    this.producto = producto;
  }

  public double getPrecio() {
    return precio;
  }

  public void setPrecio(double precio) {
    this.precio = precio;
  }

  public int getCantidad() {
    return cantidad;
  }

  public void setCantidad(int cantidad) {
```

```
    this.cantidad = cantidad;
  }

  @Override
  public String toString() {
    return this.producto + " PVP: " + String.format("%.2f", this.precio)
      + " Unidades: " + this.cantidad
      + " Subtotal: " + String.format("%.2f", this.precio * this.cantidad);
  }

}
```

Fichero: Carrito.java

```
import java.util.ArrayList;

public class Carrito {

  ArrayList<Elemento> a = new ArrayList<>();

  public void agrega(Elemento e) {
    a.add(e);
  }

  public int numeroDeElementos() {
    return a.size();
  }

  public double importeTotal() {
    double total = 0;
    for (Elemento e : a) {
      total += e.getPrecio() * e.getCantidad();
    }
    return total;
  }

  @Override
  public String toString() {
    String pinta = "Contenido del carrito\n=====================\n";
    for (Elemento e : a) {
      pinta += e + "\n";
    }
    return pinta;
  }
}
```

Fichero: S10Ejercicio17.java

```
public class S10Ejercicio17 {

  public static void main(String[] args) {
    Carrito miCarrito = new Carrito();
    miCarrito.agrega(new Elemento("Tarjeta SD 64Gb", 19.95, 2));
    miCarrito.agrega(new Elemento("Canon EOS 2000D", 449, 1));
    System.out.println(miCarrito);
    System.out.print("Hay " + miCarrito.numeroDeElementos());
    System.out.println(" productos en la cesta.");
    System.out.println("El total asciende a "
      + String.format("%.2f", miCarrito.importeTotal()) + " euros");
    System.out.println("\nContinúa la compra...\n");
    miCarrito.agrega(new Elemento("Samsung Galaxy Tab", 199, 3));
    miCarrito.agrega(new Elemento("Tarjeta SD 64Gb", 19.95, 1));
    System.out.println(miCarrito);
    System.out.print("Ahora hay " + miCarrito.numeroDeElementos());
    System.out.println(" productos en la cesta.");
    System.out.println("El total asciende a "
      + String.format("%.2f", miCarrito.importeTotal()) + " euros");
  }

}
```

Ejercicio 18

Fichero: Elemento.java

```
public class Elemento {

  private String producto;
  private double precio;
  private int cantidad;

  public Elemento(String producto, double precio, int cantidad) {
    this.producto = producto;
    this.precio = precio;
    this.cantidad = cantidad;
  }

  public String getProducto() {
    return producto;
  }
```

```
  public void setProducto(String producto) {
    this.producto = producto;
  }

  public double getPrecio() {
    return precio;
  }

  public void setPrecio(double precio) {
    this.precio = precio;
  }

  public int getCantidad() {
    return cantidad;
  }

  public void setCantidad(int cantidad) {
    this.cantidad = cantidad;
  }

  @Override
  public String toString() {
    return this.producto + " PVP: " + String.format("%.2f", this.precio)
      + " Unidades: " + this.cantidad
      + " Subtotal: " + String.format("%.2f", this.precio * this.cantidad);
  }

}
```

Fichero: Carrito.java

```
import java.util.ArrayList;

public class Carrito {

  ArrayList<Elemento> a = new ArrayList<>();

  public void agrega(Elemento e) {
    boolean encontrado = false;

    for (Elemento elemento : a) {

      if (elemento.getProducto().equals(e.getProducto())) {
        elemento.setCantidad(elemento.getCantidad() + e.getCantidad());
        encontrado = true;
      }
```

```
    }
    if (!encontrado) {
      a.add(e);
    }
  }

  public int numeroDeElementos() {
    return a.size();
  }

  public double importeTotal() {
    double total = 0;
    for (Elemento e : a) {

      total += e.getPrecio() * e.getCantidad();

    }
    return total;
  }

  @Override
  public String toString() {
    String pinta = "Contenido del carrito\n=====================\n";
    for (Elemento e : a) {
      pinta += e + "\n";
    }
    return pinta;
  }
}
```

Fichero: S10Ejercicio18.java

```
public class S10Ejercicio18 {

  public static void main(String[] args) {
    Carrito miCarrito = new Carrito();
    miCarrito.agrega(new Elemento("Tarjeta SD 64Gb", 19.95, 2));
    miCarrito.agrega(new Elemento("Canon EOS 2000D", 449, 1));
    System.out.println(miCarrito);
    System.out.print("Hay " + miCarrito.numeroDeElementos());
    System.out.println(" productos en la cesta.");
    System.out.println("El total asciende a "
      + String.format("%.2f", miCarrito.importeTotal()) + " euros");
    System.out.println("\nContinúa la compra...\n");
    miCarrito.agrega(new Elemento("Samsung Galaxy Tab", 199, 3));
    miCarrito.agrega(new Elemento("Tarjeta SD 64Gb", 19.95, 1));
```

```
    System.out.println(miCarrito);
    System.out.print("Ahora hay " + miCarrito.numeroDeElementos());
    System.out.println(" productos en la cesta.");
    System.out.println("El total asciende a "
      + String.format("%.2f", miCarrito.importeTotal()) + " euros");
  }

}
```

Ejercicio 19

Fichero: S10Ejercicio19.java

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;

public class S10Ejercicio19 {

  public static void main(String[] args) {
    HashMap<String, String> diccionario = new HashMap<String, String>();
    diccionario.put("caliente", "hot");
    diccionario.put("rojo", "red");
    diccionario.put("ardiente", "hot");
    diccionario.put("verde", "green");
    diccionario.put("agujetas", "stiff");
    diccionario.put("abrasador", "hot");
    diccionario.put("hierro", "iron");
    diccionario.put("grande", "big");

    Scanner s = new Scanner(System.in);

    String palabraIntroducida = "";

    do {
      System.out.print("\nIntroduzca una palabra y le daré los sinónimos: ");
      palabraIntroducida = s.nextLine();

      if (!palabraIntroducida.equals("salir")) {
        // Comprueba si la palabra existe en el diccionario
        if (!diccionario.containsKey(palabraIntroducida)) {
          System.out.print("No conozco esa palabra");
          // Comprueba si tiene sinónimos
        } else if (!tieneSinonimos(palabraIntroducida, diccionario)) {
```

```
          System.out.print("No conozco sinónimos de esa palabra");
          // Muestra los sinónimos
        } else {
          String significado = diccionario.get(palabraIntroducida);
          System.out.print("Sinónimos de " + palabraIntroducida + ": ");

          ArrayList<String> sinonimos = new ArrayList<>();

          for (Map.Entry entrada : diccionario.entrySet()) {
            if (!entrada.getKey().equals(palabraIntroducida)
                    && entrada.getValue().equals(significado)) {
              sinonimos.add((String) entrada.getKey());
            }
          }

          muestraLista(sinonimos);
        }
      }
    } while (!palabraIntroducida.equals("salir"));
  }

  /**
   * Me dice si una palabra tiene sinónimos dentro de un diccionario.
   *
   * @param palabra
   * @param d
   * @return
   */
  public static boolean tieneSinonimos(String palabra, HashMap<String, String> d) {
    String significado = d.get(palabra);

    int contador = 0;

    for (Map.Entry entrada : d.entrySet()) {
      if (entrada.getValue().equals(significado)) {
        contador++;
      }
    }

    return contador > 1;
  }

  public static void muestraLista(ArrayList<String> lista) {
    for (int i = 0; i < lista.size(); i++) {
      System.out.print(lista.get(i));
      if (i < lista.size() - 1) {
```

```
        System.out.print(", ");
      }
    }
  }
}
```

Ejercicio 20

Fichero: S10Ejercicio20.java

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;

public class S10Ejercicio20 {

  public static void main(String[] args) {
    HashMap<String, String> diccionario = new HashMap<String, String>();
    diccionario.put("caliente", "hot");
    diccionario.put("rojo", "red");
    diccionario.put("ardiente", "hot");
    diccionario.put("verde", "green");
    diccionario.put("agujetas", "stiff");
    diccionario.put("abrasador", "hot");
    diccionario.put("hierro", "iron");
    diccionario.put("grande", "big");

    Scanner s = new Scanner(System.in);

    String palabraIntroducida = "";

    do {
      System.out.print("Introduzca una palabra y le daré los sinónimos: ");
      palabraIntroducida = s.nextLine();

      if (!palabraIntroducida.equals("salir")) {
        // Comprueba si la palabra existe en el diccionario
        if (!diccionario.containsKey(palabraIntroducida)) {
          System.out.print("No conozco esa palabra ¿quiere añadirla al diccionario? (s/n):");
          if (s.nextLine().equals("s")) {
            System.out.print("Introduzca la traducción de " + palabraIntroducida + " en inglés\
: ");
            String traduccion = s.nextLine();
            diccionario.put(palabraIntroducida, traduccion);
```

```
        }
      // Comprueba si tiene sinónimos
      } else if (!tieneSinonimos(palabraIntroducida, diccionario)) {
        System.out.print("No conozco sinónimos de esa palabra ¿quiere añadir alguno? (s/n): \
");
        if (s.nextLine().equals("s")) {
          System.out.print("Introduzca un sinónimo de " + palabraIntroducida + ": ");
          String sinonimo = s.nextLine();
          diccionario.put(sinonimo, diccionario.get(palabraIntroducida));
          System.out.println("Gracias por enseñarme nuevos sinónimos.");
        }
      } else {
        // Muestra los sinónimos
        String significado = diccionario.get(palabraIntroducida);
        System.out.print("Sinónimos de " + palabraIntroducida + ": ");

        ArrayList<String> sinonimos = new ArrayList<>();

        for (Map.Entry entrada : diccionario.entrySet()) {
          if (!entrada.getKey().equals(palabraIntroducida)
                  && entrada.getValue().equals(significado)) {
            sinonimos.add((String) entrada.getKey());
          }
        }

        muestraLista(sinonimos);
      }
    }
  } while (!palabraIntroducida.equals("salir"));
} // main

/**
 * Me dice si una palabra tiene sinónimos dentro de un diccionario.
 *
 * @param palabra
 * @param d
 * @return
 */
public static boolean tieneSinonimos(String palabra, HashMap<String, String> d) {
  String significado = d.get(palabra);

  int contador = 0;

  for (Map.Entry entrada : d.entrySet()) {
    if (entrada.getValue().equals(significado)) {
      contador++;
```

```
      }
    }

    return contador > 1;
  }

  public static void muestraLista(ArrayList<String> lista) {
    for (int i = 0; i < lista.size(); i++) {
      System.out.print(lista.get(i));
      if (i < lista.size() - 1) {
        System.out.print(", ");
      }
    }
    System.out.println("");
  }

}
```

Ejercicio 21

Fichero: S10Ejercicio21.java

```
import java.util.HashMap;
import java.util.Scanner;

public class S10Ejercicio21 {

  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);

    HashMap<String, String> habitat = new HashMap<>();
    HashMap<String, String> alimentacion = new HashMap<>();

    habitat.put("rana", "En los trópicos y cerca de las zonas húmedas y acuáticas.");
    habitat.put("salamandra", "Ecosistemas húmedos.");
    habitat.put("sapo", "En cualquier sitio salvo el desierto y la Antártida.");
    habitat.put("tritón", "América y África.");
    alimentacion.put("rana", "Larvas e insectos.");
    alimentacion.put("salamandra", "Pequeños crustáceos e insectos.");
    alimentacion.put("sapo", "Insectos, lombrices y pequeños roedores.");
    alimentacion.put("tritón", "Insectos.");


    System.out.print("Introduzca el tipo de anfibio: ");
    String anfibio = s.nextLine();
```

```
    if (!habitat.containsKey(anfibio)) {
      System.out.println("Ese tipo de anfibio no existe.");
    } else {
      System.out.println("Hábitat: " + habitat.get(anfibio));
      System.out.println("Alimentación: " + alimentacion.get(anfibio));
    }
  }

}
```

Ejercicio 22

Fichero: Banco.java

```
public class Banco {

  public static void main(String[] args) {
    CuentaCorriente cuenta1 = new CuentaCorriente();
    CuentaCorriente cuenta2 = new CuentaCorriente(1500);
    CuentaCorriente cuenta3 = new CuentaCorriente(6000);
    cuenta1.ingreso(2000);
    cuenta1.cargo(600);
    cuenta3.ingreso(75);
    cuenta1.cargo(55);
    cuenta2.transferencia(cuenta1, 100);
    cuenta1.transferencia(cuenta3, 250);
    cuenta3.transferencia(cuenta1, 22);
    cuenta1.movimientos();
  }

}
```

Fichero: CuentaCorriente.java

```
import java.util.ArrayList;

class CuentaCorriente {

  private String numero = "";
  private double saldo;
  private ArrayList<String> movimientos = new ArrayList<>();

  public CuentaCorriente() {
```

```
    this.generaNumero();
  }

  public CuentaCorriente(double saldo) {
    this.generaNumero();
    this.saldo = saldo;
  }

  private void generaNumero() {
    for (int i = 0; i < 10; i++) {
      numero += (int) (Math.random() * 10);
    }
  }

  @Override
  public String toString() {
    return "Número de cta: " + numero + " Saldo: " + String.format("%.2f", saldo) + " €";
  }

  void ingreso(int dinero) {
    saldo += dinero;
    movimientos.add("Ingreso de " + dinero + " €  Saldo: " + String.format("%.2f", saldo) + " \
€");
  }

  void cargo(int dinero) {
    saldo -= dinero;
    movimientos.add("Cargo de " + dinero + " €  Saldo: " + String.format("%.2f", saldo) + " €"\
);
  }

  void transferencia(CuentaCorriente cuentaDestino, int dinero) {
    saldo -= dinero;
    cuentaDestino.saldo += dinero;
    movimientos.add("Transf. emitida de " + dinero + " € a la cuenta "
            + cuentaDestino.numero + "  Saldo " + String.format("%.2f", saldo) + " €");
    cuentaDestino.movimientos.add("Transf. recibida de " + dinero
            + " € de la cuenta " + cuentaDestino.numero + "  Saldo " + String.format("%.2f", c\
uentaDestino.saldo) + " €");
  }

  void movimientos() {
    System.out.println("Movimientos de la cuenta " + numero);
    System.out.println("-----------------------------------");
    for (String movimiento : movimientos) {
      System.out.println(movimiento);
```

```
    }
  }
}
```

Ejercicio 23

Fichero: S10Ejercicio23.java

```
import java.util.ArrayList;

public class S10Ejercicio23 {

  public static void main(String[] args) {
    ArrayList<Figura> figuras = new ArrayList<>();

    figuras.add(new Figura("Dama", 9, 1));
    figuras.add(new Figura("Torre", 5, 2));
    figuras.add(new Figura("Alfil", 3, 2));
    figuras.add(new Figura("Caballo", 2, 2));
    figuras.add(new Figura("Peón", 1, 8));

    System.out.println("Fichas capturadas por el jugador:");

    int capturasTotales = (int) (Math.random() * 16);
    int contadorCapturas = 0;
    int peonesTotales = 0;

    do {
      // Genera una captura aleatoria
      Figura figuraAux = figuras.get((int) (Math.random() * figuras.size()));
      if (figuraAux.esCapturable()) {
        figuraAux.captura();
        System.out.println(figuraAux);
        contadorCapturas++;
        peonesTotales += figuraAux.getValor();
      }
    } while (contadorCapturas < capturasTotales);

    System.out.println("Puntos totales: " + peonesTotales);
  }

}
```

Fichero: Figura.java

```
class Figura {
  private String nombre;
  private int valor; // valor en número de peones
  private int cantidad; // cantidad

  public Figura(String nombre, int valor, int cantidad) {
    this.nombre = nombre;
    this.valor = valor;
    this.cantidad = cantidad;
  }

  public boolean esCapturable() {
    return this.cantidad > 0;
  }

  public void captura() {
    this.cantidad--;
  }

  @Override
  public String toString() {
    return nombre + " (" + valor + (valor == 1 ? " peón)" : " peones)");
  }

  public int getValor() {
    return valor;
  }

}
```

Ficheros de texto y paso de parámetros por línea de comandos

Ejercicio 1

Fichero: S11Ejercicio01.java

```
/**
 * 1. Escribe un programa que guarde en un fichero con nombre <code>
 *    primos.dat</code> los números primos que hay entre 1 y 500.
 *
 * @author Luis José Sánchez
 */
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;

class S11Ejercicio01 {
  public static void main(String[] args) {
    try {
      BufferedWriter bw = new BufferedWriter(new FileWriter("primos.dat"));

      for (int i = 1; i < 501; i++) {
        if (esPrimo(i)) {
          bw.write(String.valueOf(i) + "\n");
        }
      }
      bw.close();

    } catch (IOException ioe) {
        System.out.println("Error de escritura.");
    }
  }

  /**
   * Devuelve verdadero si el número que se pasa como parámetro es primo y falso
   * en caso contrario.
   * <p>
   * Un número es primo cuando es divisible únicamente entre el mismo y entre 1.
   *
   * @param x número del que se quiere saber si es primo
   * @return  verdadero si el número que se pasa como parámetro es primo y falso
   *          en caso contrario
   */
  public static boolean esPrimo(int x) {
```

```
    for (int i = 2; i < x; i++) {
      if ((x % i) == 0) {
        return false;
      }
    }
    return true;
  }
}
```

Ejercicio 2

Fichero: S11Ejercicio02.java

```
/**
 * 2. Realiza un programa que lea el fichero creado en el ejercicio anterior y
 *    que muestre los números por pantalla.
 *
 * @author Luis José Sánchez
 */
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

class S11Ejercicio02 {
  public static void main(String[] args) {
    try {
      BufferedReader br = new BufferedReader(new FileReader("primos.dat"));

      String linea = "";
      while (linea != null) {
        System.out.print(linea + " ");
        linea = br.readLine();
      }
      System.out.println();
      br.close();

    } catch (IOException e) {
      System.out.println("Error de lectura.");
    }
  }
}
```

Ejercicio 3

Fichero: S11Ejercicio03.java

```
/**
 * 3. Escribe un programa que guarde en un fichero el contenido de otros dos
 *    ficheros, de tal forma que en el fichero resultante aparezcan las líneas
 *    de los primeros dos ficheros mezcladas, es decir, la primera línea será
 *    del primer fichero, la segunda será del segundo fichero, la tercera será
 *    la siguiente del primer fichero, etc.
 *    Los nombres de los dos ficheros origen y el nombre del fichero destino se
 *    deben pasar como argumentos en la línea de comandos.
 *    Hay que tener en cuenta que los ficheros de donde se van cogiendo las
 *    líneas pueden tener tamaños diferentes.
 *
 * @author Luis José Sánchez
 */
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

class S11Ejercicio03 {
  public static void main(String[] args) {
    if (args.length != 3) {
      System.out.println("Uso del programa: S11Ejercicio03 FICHERO1 FICHERO2 MEZCLA");
      System.exit(-1); // sale del programa
    }
    try {
      BufferedReader bf1 = new BufferedReader(new FileReader(args[0]));
      BufferedReader bf2 = new BufferedReader(new FileReader(args[1]));
      BufferedWriter bw = new BufferedWriter(new FileWriter(args[2]));

      String linea1 = "";
      String linea2 = "";

      while ( (linea1 != null) || (linea2 != null) ) {
        linea1 = bf1.readLine();
        linea2 = bf2.readLine();

        if (linea1 != null) {
          bw.write(linea1 + "\n");
        }

        if (linea2 != null) {
          bw.write(linea2 + "\n");
        }
      }
```

```
      bf1.close();
      bf2.close();
      bw.close();

    } catch (IOException ioe) {
      System.out.println("Se ha producido un error de lectura/escritura");
      System.err.println(ioe.getMessage());
    }
  }
}
```

Ejercicio 4

Fichero: S11Ejercicio04.java

```
/**
 * 4. Realiza un programa que sea capaz de ordenar alfabéticamente las palabras
 *    contenidas en un fichero de texto. El nombre del fichero que contiene las
 *    palabras se debe pasar como argumento en la línea de comandos. El nombre
 *    del fichero resultado debe ser el mismo que el original añadiendo la
 *    coletilla <code>sort</code>, por ejemplo <code>palabras_sort.txt</code>.
 *    Suponemos que cada palabra ocupa una línea.
 *
 * @author Luis José Sánchez
 */
import java.util.Collections;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Vector;

class S11Ejercicio04 {
  public static void main(String[] args) {

    if (args.length != 1) {
      System.out.println("Uso del programa: S11Ejercicio04 FICHERO.TXT");
      System.exit(-1);
    }

    try {
      BufferedReader br = new BufferedReader(new FileReader(args[0]));
      int l = args[0].length();
      String nombre = args[0].substring(0, l - 4);
```

```
      String extension = args[0].substring(l - 4, l);

      BufferedWriter bw = new BufferedWriter(new FileWriter(nombre + "_sort" + extension));

      Vector<String> v = new Vector<String>();

      String linea = "";
      while (linea != null) {
        v.addElement(linea);
        linea = br.readLine();
      }
      br.close();

      v.removeElementAt(0);
      Collections.sort(v);

      for (String s: v) {
        bw.write(s + "\n");
      }

      bw.close();
    } catch (IOException ioe) {
      System.err.println(ioe.getMessage());
    }
  }
}
```

Ejercicio 5

Fichero: S11Ejercicio05.java

```
/**
 * 5. Escribe un programa capaz de quitar los comentarios de un programa de
 *    Java. Se utilizaría de la siguiente manera:
 *    <p>
 *    <code>quita_commentarios PROGRAMA_ORIGINAL PROGRAMA_LIMPIO</code>
 *    <p>
 *    Por ejemplo:
 *    <p>
 *    <code>quita_comentarios hola.java holav2.java</code>
 *    <p>
 *    crea un fichero con nombre <code>holav2.java</code> que contiene el código
 *    de <code>hola.java</code> pero sin los comentarios.
 *
 * @author Luis José Sánchez
```

```
 */
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

class FichEjercicio05 {
  public static void main(String[] args) {

    if (args.length != 2) {
      System.out.println("Uso del programa: S11Ejercicio05 PROGRAMA_ORIGINAL PROGRAMA_LIMPIO");
      System.exit(-1);
    }

    try {
      BufferedReader br = new BufferedReader(new FileReader(args[0]));
      BufferedWriter bw = new BufferedWriter(new FileWriter(args[1]));

      String lineaOrigen = "";
      String lineaDestino = "";
      boolean comentarioDeBloque = false;
      int i = 0;

      while ((lineaOrigen = br.readLine()) != null) {

        int l = lineaOrigen.length();
        lineaDestino = lineaOrigen;

        // Detecta inicio de comentario de bloque
        if ((i = lineaOrigen.indexOf("/*")) != -1) {
          comentarioDeBloque = true;
          lineaDestino = lineaOrigen.substring(0, i);
          lineaOrigen = lineaDestino;
        }

        // Detecta fin de comentario de bloque
        if ((i = lineaOrigen.indexOf("*/")) != -1) {
          comentarioDeBloque = false;
          lineaDestino = lineaOrigen.substring(i + 2, l);
        }
        // Detecta comentario de línea
        if (((i = lineaOrigen.indexOf("//")) != -1) && !comentarioDeBloque) {
          lineaDestino = lineaOrigen.substring(0, i);
        }
```

```
          if (!comentarioDeBloque) {
            bw.write(lineaDestino + "\n");
          }
        }

        br.close();
        bw.close();

      } catch (IOException ioe) {
        System.err.println(ioe.getMessage());
      }
    }
  }
```

Ejercicio 6

Fichero: S11Ejercicio06.java

```
/**
 * 6. Realiza un programa que diga cuántas ocurrencias de una palabra hay en un
 *    fichero. Tanto el nombre del fichero como la palabra se deben pasar como
 *    argumentos en la línea de comandos.
 *
 * @author Luis José Sánchez
 */
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

class FichEjercicio06 {

  public static void main(String[] args) {
    if (args.length != 2) {
      System.out.println("Uso del programa: FichEjercicio06 FICHERO PALABRA");
      System.exit(-1);
    }

    try {
      BufferedReader br = new BufferedReader(new FileReader(args[0]));
      String palabra = args[1];
      String linea = "";
      int i = 0;
      int repeticiones = 0;

      while ((linea = br.readLine()) != null) {
```

```
        while ((i = linea.indexOf(palabra)) != -1) {
          linea = linea.substring(i + palabra.length(), linea.length());
          repeticiones++;
        }
      }

      br.close();

      System.out.println("La palabra " + palabra + " aparece " + repeticiones + " veces en el \
fichero.");

    } catch (IOException ioe) {
      System.err.println(ioe.getMessage());
    }
  }
}
```

Aplicaciones web en Java (JSP)

Ejercicio 1

Fichero: index.jsp

```
<%--
  1. Crea una aplicación web en Java que muestre tu nombre y tus datos
  personales por pantalla. La página principal debe ser un archivo con la
  extensión jsp. Comprueba que se lanzan bien el servidor y el navegador
  web. Observa los mensajes que da el servidor. Fíjate en la dirección que
  aparece en la barra de direcciones del navegador.
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Relación 12 - Ejercicio 1</title>
  </head>
  <body>
    <h2>Luis José Sánchez González</h2>
    C/ Larios, 7 Málaga - (España)</br>
    Tlf: 952 555 666
  </body>
</html>
```

Ejercicio 2

Fichero: index.jsp

```
<%--
  2. Mejora el programa anterior de tal forma que la apariencia de la página
  web que muestra el navegador luzca más bonita mediante la utilización de
  CSS (utiliza siempre fcheros independientes para CSS para no mezclarlo
  con HTML).
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

```
    <title>Relación 12 - Ejercicio 2</title>
    <link rel="stylesheet" type="text/css" href="estilos.css">
  </head>
  <body>
    <h2>Luis José Sánchez González</h2>
    C/ Larios, 7 Málaga - (España)</br>
    Tlf: 952 555 666
  </body>
</html>
```

Fichero: estilos.css

```
root {
  display: block;
}

body {
  color:brown;
  background:beige;
  font-family:sans-serif;
  font-size: 12px;
}

h2 {
  color:cadetblue;
  font-size: 18px;
}
```

Ejercicio 3

Fichero: index.jsp

```
<%--
  3. Escribe una aplicación que pida tu nombre. A continuación mostrará "Hola"
  seguido del nombre introducido. El nombre se deberá recoger mediante un
  formulario.
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Relación 12 - Ejercicio 3</title>
```

```
    <link rel="stylesheet" type="text/css" href="estilos.css">
  </head>
  <body>
    <form method="post" action="saludo.jsp">
      ¿Cómo te llamas?
      <input type="text" name="nombre">
      <input type="submit" value="OK">
    </form>
  </body>
</html>
```

Fichero: saludo.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Saludo</title>
  </head>
  <body>
    <% request.setCharacterEncoding("UTF-8"); %>
    Hola <% out.print(request.getParameter("nombre")); %>
  </body>
</html>
```

Ejercicio 4

Fichero: index.jsp

```
<%--
  4. Realiza una aplicación que calcule la media de tres notas.
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Relación 12 - Ejercicio 4</title>
  </head>
  <body>
    <h2>Calcula la media de tres notas</h2>
    <form method="post" action="media.jsp">
```

```
      Nota 1: <input type="number" min="0" max="10" step="0.01" name="nota1"></br>
      Nota 2: <input type="number" min="0" max="10" step="0.01" name="nota2"></br>
      Nota 3: <input type="number" min="0" max="10" step="0.01" name="nota3"></br></br>
      <input type="submit" value="Aceptar">
    </form>
  </body>
</html>
```

Fichero: media.jsp

```
<%@page import="java.text.DecimalFormat"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Media</title>
  </head>
  <body>
    <%
      double n1 = Double.valueOf(request.getParameter("nota1"));
      double n2 = Double.valueOf(request.getParameter("nota2"));
      double n3 = Double.valueOf(request.getParameter("nota3"));
      double media = (n1 + n2 +n3) / 3;
      DecimalFormat dosDecimales = new DecimalFormat("0.00");
      out.println("La media es " + dosDecimales.format(media));
    %>
  </body>
</html>
```

Ejercicio 5

Fichero: eurosapesetas.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Euros a pesetas</title>
  </head>
  <body>
    <% Double e = Double.parseDouble(request.getParameter("euros")); %>
    <% out.print(e); %> euros son <% out.println(Math.round(e * 166.386)); %> pesetas.
  </body>
</html>
```

Fichero: index.jsp

```
<%--
  5. Realiza un conversor de euros a pesetas.
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Relación 12 - Ejercicio 5</title>
  </head>
  <body>
    <h2>Conversor de euros a pesetas</h2>
    <form method="post" action="eurosapesetas.jsp">
      Euros: <input type="number" min="0" step="0.01" name="euros"></br></br>
      <input type="submit" value="Aceptar">
    </form>
  </body>
</html>
```

Ejercicio 6

Fichero: pesetasaeuros.jsp

```
<%@page import="java.text.DecimalFormat"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Relación 12 - Ejercicio 6</title>
  </head>
  <body>
    <p>
      <%
        int p = Integer.parseInt(request.getParameter("pesetas"));
        DecimalFormat formatoEuros = new DecimalFormat("0.00");
        out.print(p + " pesetas son " + formatoEuros.format(p / 166.386) + " euros.");
      %>
    </p>
  </body>
</html>
```

Fichero: index.jsp

```
<%--
  6. Realiza un conversor de pesetas a euros.
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Relación 12 - Ejercicio 6</title>
  </head>
  <body>
    <h2>Conversor de pesetas a euros</h2>
    <form method="post" action="pesetasaeuros.jsp">
      Pesetas: <input type="number" min="0" name="pesetas"></br></br>
      <input type="submit" value="Aceptar">
    </form>
  </body>
</html>
```

Ejercicio 7

Fichero: index.jsp

```
<%--
    7. Combina las dos aplicaciones anteriores en una sola de tal forma que en la
       página principal aparezcan dos formularios y se pueda elegir pasar de
       euros a pesetas o de pesetas a euros según dónde introduzcamos el valor y
       el botón que pulsemos. Adorna la página con alguna foto o dibujo.
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Relación 12 - Ejercicio 7</title>
    <link href="estilos.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <h1>conversor de moneda</h1>
    <form class="dinero" method="post" action="conversor.jsp">
      cantidad</br><input type="number" min="0" step="0.01" name="cantidad"></br></br>
      <input type="radio" name="cambio" value="eurosEnPesetas" checked="checked"> euros -> pes\
etas</br></br>
```

```
      <input type="radio" name="cambio" value="pesetasEnEuros"> pesetas -> euros</br></br>
      <input type="submit" value="Convertir">
    </form>
  </body>
</html>
```

Fichero: estilos.css

```
root {
  display: block;
}

body {
  background-image: url(500euros.jpg);
  background-repeat: repeat;
  font-family: Arial, sans-serif;
}

h1 {
  text-align: center;
  color: gold;
  text-shadow: #242424 2px 2px 2px;
}

h2 {
  text-align: center;
  padding-top: 32px;
  font-size: 18px;
}
.dinero {
  width: 400px;
  height: 200px;
  background-color: gold;
  margin: auto;
  box-shadow: #242424 4px 4px 4px;
  background-image: url(euros-reverso.png);
  background-repeat: no-repeat;
  background-position: bottom right;
}

form {
  padding: 20px;
  text-align: center;
}
```

Fichero: conversor.jsp

```
<%@page import="java.text.DecimalFormat"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Relación 12 - Ejercicio 7</title>
    <link href="estilos.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <div class="dinero">
      <h2>
        <%
          Double c = Double.parseDouble(request.getParameter("cantidad"));
          DecimalFormat formatoEuros = new DecimalFormat("0.00");

          if (request.getParameter("cambio").equals("eurosEnPesetas")) {
            out.print(formatoEuros.format(c) + " euros son " + Math.round(c * 166.386) + " pes\
etas.");
          } else {
            out.print(Math.round(c) + " pesetas son " + formatoEuros.format(c / 166.386) + " e\
uros.");
          }
        %>
      </h2>
    </div>
  </body>
</html>
```

Ejercicio 8

Fichero: index.jsp

```
<%--
  8. Realiza una aplicación que pida un número y que luego muestre la tabla de
     multiplicar de ese número. El resultado se debe mostrar en una tabla
     (<table> en HTML).
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Relación 12 - Ejercicio 8</title>
```

```
    <link href="estilos.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <h1>Tabla de multiplicar</h1>
    <form class="tabla" method="post" action="tabla.jsp">
      Introduzca un número: <input type="number" name="numero">
      <input type="submit" value="Aceptar">
    </form>
  </body>
</html>
```

Fichero: tabla.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Relación 12 - Ejercicio 8</title>
    <link href="estilos.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <% int n = Integer.parseInt(request.getParameter("numero")); %>
    <h1>Tabla de multiplicar</h1>
    <table>
      <%
        for(int i = 0; i < 11; i++) {
          out.println("<tr><td>" + i + " x " + n + "</td><td> = </td><td>" + i * n + "</td></t\
r>");
        }
      %>
    </table>
  </body>
</html>
```

Fichero: estilos.css

```
root {
  display: block;
}

body {
  background-color: #ddeedd;
  font-family: Arial, sans-serif;
}

h1 {
  text-align: center;
  color: aquamarine;
  text-shadow: #242424 2px 2px 2px;
}

h2 {
  text-align: center;
  padding-top: 32px;
  font-size: 18px;
}

table {
  background-color: aquamarine;
  margin: auto;
  box-shadow: #242424 4px 4px 4px;
  padding: 20px;
  text-align: center;
}
```

Ejercicio 9

Fichero: piramide.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Relación 12 - Ejercicio 9</title>
  </head>
  <body>
    <%
      int alturaTotal = Integer.parseInt(request.getParameter("altura"));
      int altura = 1;
      int i = 0;
```

```
      int espacios = alturaTotal - 1;

      while (altura <= alturaTotal) {

        // inserta espacios (imagenes en blanco)
        for (i = 1; i <= espacios; i++) {
          out.print("<img src=\"blanco.jpg\">");
        }

        // pinta la línea
        for (i = 1; i < altura * 2; i++) {
          out.print("<img src=\"gatoloco.jpg\">");
        }

        out.println("</br>");

        altura++;
        espacios--;
      } // while
     %>
  </body>
</html>
```

Fichero: index.jsp

```
<%--
  9. Realiza una aplicación que pinte una pirámide. La altura se pedirá en un
  formulario. La pirámide estará hecha de bolitas, ladrillos o cualquier otra
  imagen.
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Relación 12 - Ejercicio 9</title>
    <link href="estilos.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <h1>Pinta una pirámide</h1>
    <form method="get" action="piramide.jsp">
      Introduzca la altura de la pirámide: </br><input type="number" min="1" max="10" name="al\
tura"></br></br>
      <input type="submit" value="Aceptar">
    </form>
```

```
  </body>
</html>
```

Fichero: estilos.css

```
root {
  display: block;
}

body {
  font-family: Arial, sans-serif;
  background-color: #abcdef;
}

h1 {
  text-align: center;
  color: gold;
  text-shadow: #242424 2px 2px 2px;
}

h2 {
  text-align: center;
  padding-top: 32px;
  font-size: 18px;
}

form {
  background-color: gold;
  width: 400px;
  margin: auto;
  padding: 60px;
  box-shadow: #242424 4px 4px 4px;
  background-image: url(euros-reverso.png);
  background-repeat: no-repeat;
  background-position: bottom right;
  text-align: center;
}
```

Ejercicio 10

Fichero: index.jsp

```
<%--
  10. Realiza un cuestionario con 10 preguntas tipo test sobre las asignaturas
      que se imparten en el curso. Cada pregunta acertada sumará un punto. El
      programa mostrará al final la calificación obtenida. Pásale el cuestionario
      a tus compañeros y pídeles que lo hagan para ver qué tal andan de
      conocimientos en las diferentes asignaturas del curso. Utiliza radio
      buttons en las preguntas del cuestionario.
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Relación 12 - Ejercicio 10</title>
    <link href="estilos.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <h1>Cuestionario 1º DAW</h1>
    <form method="post" action="calcula_nota.jsp">
      1. ¿Cuál de los siguientes tipos de datos de Java tiene más precisión?</br>
      <input type="radio" name="p1" value="0"> int</input></br>
      <input type="radio" name="p1" value="1"> double</input></br>
      <input type="radio" name="p1" value="0"> float</input></br></br>

      2. ¿Cuál es el lenguaje que se utiliza para hacer consultas en las bases de datos?</br>
      <input type="radio" name="p2" value="0"> XML</input></br>
      <input type="radio" name="p2" value="0"> SELECT</input></br>
      <input type="radio" name="p2" value="1"> SQL</input></br></br>

      3. Para insertar un hiperenlace en una página se utiliza la etiqueta...</br>
      <input type="radio" name="p3" value="0"> href</input></br>
      <input type="radio" name="p3" value="1"> a</input></br>
      <input type="radio" name="p3" value="0"> link</input></br></br>

      4. ¿En qué directorio se encuentran los archivos de configuración de Linux?</br>
      <input type="radio" name="p4" value="1"> /etc</input></br>
      <input type="radio" name="p4" value="0"> /config</input></br>
      <input type="radio" name="p4" value="0"> /cfg</input></br></br>

      5. ¿Cuál de las siguientes memorias es volátil?</br>
      <input type="radio" name="p5" value="1"> RAM</input></br>
      <input type="radio" name="p5" value="0"> EPROM</input></br>
      <input type="radio" name="p5" value="0"> ROM</input></br></br>

      6. En Java, para definir una clase como subclase de otra se utiliza...</br>
```

```
      <input type="radio" name="p6" value="1"> extends</input></br>
      <input type="radio" name="p6" value="0"> inherit</input></br>
      <input type="radio" name="p6" value="0"> subclass</input></br></br>

      7. ¿Java soporta la herencia múltiple?</br>
      <input type="radio" name="p7" value="0"> Sí</input></br>
      <input type="radio" name="p7" value="1"> No</input></br>
      <input type="radio" name="p7" value="0"> A veces</input></br></br>

      8. ¿Qué significan las siglas CSS?</br>
      <input type="radio" name="p8" value="0"> Computer Style Sheets</input></br>
      <input type="radio" name="p8" value="0"> Creative Style Sheets</input></br>
      <input type="radio" name="p8" value="1"> Cascading Style Sheets</input></br></br>

      9. ¿Qué propiedad se utiliza en CSS para cambiar el color de fondo?</br>
      <input type="radio" name="p9" value="0"> bgcolor:</input></br>
      <input type="radio" name="p9" value="0"> color:</input></br>
      <input type="radio" name="p9" value="1"> background-color:</input></br></br>

      10. ¿Cómo se muestran los hiperenlaces sin subrayar?</br>
      <input type="radio" name="p10" value="0"> a {text-decoration:no underline}</input></br>
      <input type="radio" name="p10" value="0"> a {underline:none}</input></br>
      <input type="radio" name="p10" value="1"> a {text-decoration:none}</input></br></br>

      <input type="submit" value="Evaluar cuestionario">
    </form>
  </body>
</html>
```

Fichero: estilos.css

```
root {
  display: block;
}

body {
  font-family: Arial, sans-serif;
  background-color: #bcdeff;
}

p {
  text-align: center;
}

h1 {
  text-align: center;
```

```
  color: #9abcde;
  text-shadow: #242424 2px 2px 1px;
}

h2 {
  text-align: center;
  padding-top: 32px;
  font-size: 18px;
}

form {
  background-color: #abcdef;
  width: 600px;
  margin: auto;
  padding: 60px;
  box-shadow: #242424 4px 4px 4px;
  border-radius: 10px;
  text-align: justify;
}
```

Fichero: calcula_nota.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Relación 12 - Ejercicio 10</title>
    <link href="estilos.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <h1>Cuestionario 1º DAW</h1>
    <p>
      Ha obtenido
      <%
        int puntos = 0;

        for (Integer i = 1; i < 11; i++) {
          puntos += Integer.parseInt(request.getParameter("p" + i.toString()));
        }

        out.print(puntos);
      %>
      puntos, haga clic <a href="index.jsp">aquí</a> para repetir el cuestionario.
    </p>
  </body>
```

```
</html>
```

Ejercicio 11

Fichero: calendario.jsp

```
<%--
11. Escribe una aplicación que genere el calendario de un mes. Se pedirá el
    nombre del mes, el año, el texto que queremos que aparezca sobre el
    calendario, el día de la semana en que cae el día 1 y el número de días que
    tiene el mes.
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Relación 12 - Ejercicio 11</title>
    <link href="estilos.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <h1><% out.print(request.getParameter("mes") + " " + request.getParameter("anio"));%></h1>
    <%
      int columna = 1;
      int diasTotales = Integer.parseInt(request.getParameter("diastotales"));
      int dia1 = Integer.parseInt(request.getParameter("dia1"));
    %>
    <table>
      <tr><th>lunes</th><th>martes</th><th>miércoles</th><th>jueves</th><th>viernes</th><th>sá\
bado</th><th>domingo</th></tr>
      <tr>
      <%
      for (int i = 1; i < diasTotales + dia1; i++) {
        if (i >= dia1) {
          out.print("<td>" + (i - dia1 + 1) + "</td>");
        } else {
          out.print("<td> </td>"); // las primeras celdas quedan vacías
        }
        columna++;
        if (columna == 8) { // salta a la siguiente columna
          out.println("</tr><tr>");
          columna = 1;
        }
      }
```

```
      %>
      </td></tr>
    </table>
    </form>
  </body>
</html>
```

Fichero: index.jsp

```
<%--
11. Escribe una aplicación que genere el calendario de un mes. Se pedirá el
    nombre del mes, el año, el texto que queremos que aparezca sobre el
    calendario, el día de la semana en que cae el día 1 y el número de días que
    tiene el mes.
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Relación 12 - Ejercicio 11</title>
    <link href="estilos.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <h1>Generador de calendario</h1>
    <form method="post" action="calendario.jsp">
      Mes
      <select name="mes">
        <option>enero</option>
        <option>febrero</option>
        <option>marzo</option>
        <option>abril</option>
        <option>mayo</option>
        <option>junio</option>
        <option>julio</option>
        <option>agosto</option>
        <option>septiembre</option>
        <option>octubre</option>
        <option>noviembre</option>
        <option>diciembre</option>
      </select>
      Año <input type="number" min="1" name="anio"></br></br>
      El día 1 cae en
      <select name="dia1">
        <option value="1">lunes</option>
```

```
        <option value="2">martes</option>
        <option value="3">miércoles</option>
        <option value="4">jueves</option>
        <option value="5">viernes</option>
        <option value="6">sábado</option>
        <option value="7">domingo</option>
      </select></br></br>
      Número de días que tiene el mes <input type="number" min="28" max="31" name="diastotales\
"></br></br>
      <hr>
      </br></br>
      <input type="submit" value="Aceptar">
    </form>
  </body>
</html>
```

Fichero: estilos.css

```
root {
  display: block;
}

body {
  font-family: Arial, sans-serif;
  background-color: #bcdeff;
}

p {
  text-align: center;
}

h1 {
  text-align: center;
  color: white;
  text-shadow: #242424 2px 2px 1px;
}

h2 {
  text-align: center;
  padding-top: 32px;
  font-size: 18px;
}

form {
  background-color: #abcdef;
  width: 400px;
```

```
  margin: auto;
  padding: 60px;
  box-shadow: #242424 4px 4px 4px;
  border-radius: 6px;
  text-align: center;
}

table {
  background-color: #abcdef;
  margin: auto;
  padding: 20px;
  box-shadow: #242424 4px 4px 4px;
  text-align: right;
  border: 0px;
  border-collapse: collapse;
}

th {
  background-color: #aaaadd;
  color: white;
  padding: 4px;
  margin: 2px;
}

td {
  padding: 10px;
}
```

Ejercicio 12

Fichero: calendario.jsp

```
<%@page import="java.util.Calendar"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Relación 12 - Ejercicio 12</title>
    <link href="estilos.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <%
      int mes = Integer.parseInt(request.getParameter("mes"));
      int anio = Integer.parseInt(request.getParameter("anio"));
```

```
      String[] nombreMes = {"enero", "febrero", "marzo", "abril", "mayo", "junio", "julio", "a\
gosto", "septiembre", "octubre", "noviembre", "diciembre"};

      Calendar miCalendario = Calendar.getInstance();
      miCalendario.set(anio, mes - 1, 1); // los meses van de 0 a 11

      int diasTotales = miCalendario.getActualMaximum(Calendar.DAY_OF_MONTH);

      int dia1 = miCalendario.get(Calendar.DAY_OF_WEEK);
      if (dia1 == 1) {
        dia1 = 7; // el domingo es día 1
      } else {
        dia1--;
      }

      int columna = 1;
    %>

    <h1><% out.print(nombreMes[mes - 1] + " " + anio);%></h1>

    <table>
      <tr><th>lunes</th><th>martes</th><th>miércoles</th><th>jueves</th><th>viernes</th><th>sá\
bado</th><th>domingo</th></tr>
      <tr>
      <%
      for (int i = 1; i < diasTotales + dia1; i++) {
        if (i >= dia1) {
          out.print("<td>" + (i - dia1 + 1) + "</td>");
        } else {
          out.print("<td> </td>"); // pinta los huecos que corresponden a los días del mes pas\
ado
        }
        columna++;
        if (columna == 8) { // termina la fila actual
          out.println("</tr><tr>");
          columna = 1;
        }
      }
      %>
      </td></tr>
    </table>
    </form>
  </body>
</html>
```

Fichero: index.jsp

```
<%--
12. Mejora la aplicación anterior de tal forma que no haga falta introducir el
    día de la semana en que cae el día 1 y el número de días que tiene el mes.
    El programa debe deducir estos datos del mes y el año.
    Pista: puedes usar la clase Calendar (java.util.Calendar).
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Relación 12 - Ejercicio 12</title>
    <link href="estilos.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <h1>Generador de calendario</h1>
    <form method="post" action="calendario.jsp">
      mes
      <select name="mes">
        <option value="1">enero</option>
        <option value="2">febrero</option>
        <option value="3">marzo</option>
        <option value="4">abril</option>
        <option value="5">mayo</option>
        <option value="6">junio</option>
        <option value="7">julio</option>
        <option value="8">agosto</option>
        <option value="9">septiembre</option>
        <option value="10">octubre</option>
        <option value="11">noviembre</option>
        <option value="12">diciembre</option>
      </select>
       año <input type="number" name="anio" size="4">
       <input type="submit" value="Aceptar">
    </form>
  </body>
</html>
```

Fichero: estilos.css

```
root {
  display: block;
}

body {
  font-family: Arial, sans-serif;
  background-color: #bcdeff;
}

p {
  text-align: center;
}

h1 {
  text-align: center;
  color: white;
  text-shadow: #242424 2px 2px 1px;
}

h2 {
  text-align: center;
  padding-top: 32px;
  font-size: 18px;
}

form {
  background-color: #abcdef;
  width: 460px;
  margin: auto;
  padding: 60px 10px 60px 10px;
  box-shadow: #242424 4px 4px 4px;
  border-radius: 6px;
  text-align: center;
}

table {
  background-color: #abcdef;
  margin: auto;
  padding: 20px;
  box-shadow: #242424 4px 4px 4px;
  text-align: right;
  border: 0px;
  border-collapse: collapse;
}

th {
```

```
  background-color: #aaaadd;
  color: white;
  padding: 4px;
  margin: 2px;
}

td {
  padding: 10px;
}
```

Ejercicio 13

Fichero: cuestionario.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Relación 12 - Ejercicio 13</title>
    <link href="estilos.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <h1>Test de infidelidad</h1>
    <div id="principal">
      <p>
        <%
          int puntos = 0;

          for (Integer i = 1; i < 11; i++) {
            puntos += Integer.parseInt(request.getParameter("p" + i.toString()));
          }

          if (puntos <= 10) {
            out.println("¡Enhorabuena! tu pareja parece ser totalmente fiel.");
          }

          if ((puntos >= 11) && (puntos <= 22)) {
            out.println("Quizás exista el peligro de otra persona en su vida o en su mente, au\
nque seguramente será algo sin importancia. No bajes la guardia.");
          }

          if (puntos > 22) {
            out.println("Tu pareja tiene todos los ingredientes para estar viviendo un romance\
 con otra persona. Te aconsejamos que indagues un poco más y averigües que es lo que está pasa\
```

```
ndo por su cabeza.");
          }
      %>
      <br><br>Haga clic <a href="index.jsp">aquí</a> para repetir el cuestionario.
      </p>
    </div>
  </body>
</html>
```

Fichero: index.jsp

```
<%--
  13. Pasa el test de infidelidad de programa en consola a aplicación web.
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Relación 12 - Ejercicio 13</title>
    <link href="estilos.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <h1>Test de infidelidad</h1>
    <div id="principal">
    <form method="post" action="cuestionario.jsp">
      1. Tu pareja parece estar más inquieta de lo normal sin ningún motivo aparente.</br>
      <input type="radio" name="p1" value="3"> verdadero</input></br>
      <input type="radio" name="p1" value="0"> falso</input></br></br>
      2. Ha aumentado sus gastos de vestuario</br>
      <input type="radio" name="p2" value="3"> verdadero</input></br>
      <input type="radio" name="p2" value="0"> falso</input></br></br>
      3. Ha perdido el interés que mostraba anteriormente por ti</br>
      <input type="radio" name="p3" value="3"> verdadero</input></br>
      <input type="radio" name="p3" value="0"> falso</input></br></br>
      4. Ahora se afeita y se asea con más frecuencia (si es hombre) o ahora se arregla el pel\
o y se asea con más frecuencia (si es mujer)</br>
      <input type="radio" name="p4" value="3"> verdadero</input></br>
      <input type="radio" name="p4" value="0"> falso</input></br></br>
      5. No te deja que mires la agenda de su teléfono móvil</br>
      <input type="radio" name="p5" value="3"> verdadero</input></br>
      <input type="radio" name="p5" value="0"> falso</input></br></br>
      6. A veces tiene llamadas que dice no querer contestar cuando estás tú delante</br>
      <input type="radio" name="p6" value="3"> verdadero</input></br>
      <input type="radio" name="p6" value="0"> falso</input></br></br>
```

```
      7. Últimamente se preocupa más en cuidar la línea y/o estar bronceado/a</br>
      <input type="radio" name="p7" value="3"> verdadero</input></br>
      <input type="radio" name="p7" value="0"> falso</input></br></br>
      8. Muchos días viene tarde después de trabajar porque dice tener mucho más trabajo</br>
      <input type="radio" name="p8" value="3"> verdadero</input></br>
      <input type="radio" name="p8" value="0"> falso</input></br></br>
      9. Has notado que últimamente se perfuma más</br>
      <input type="radio" name="p9" value="3"> verdadero</input></br>
      <input type="radio" name="p9" value="0"> falso</input></br></br>
      10. Se confunde y te dice que ha estado en sitios donde no ha ido contigo</br>
      <input type="radio" name="p10" value="3"> verdadero</input></br>
      <input type="radio" name="p10" value="0"> falso</input></br></br>
      <input type="submit" value="Evaluar cuestionario">
    </form>
    </div>
  </body>
</html>
```

Fichero: estilos.css

```
root {
  display: block;
}

body {
  font-family: Arial, sans-serif;
  background-color: #bcdeff;
}

p {
  text-align: center;
}

h1 {
  text-align: center;
  color: white;
  text-shadow: #242424 2px 2px 1px;
}

#principal {
  background-color: #abcdef;
  width: 600px;
  margin: auto;
  padding: 60px;
  box-shadow: #242424 4px 4px 4px;
  border-radius: 6px;
```

```
  text-align: justify;
}
```

Ejercicio 14

Fichero: fibonacci.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Relación 12 - Ejercicio 14</title>
    <link href="estilos.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <h1>Serie de Fibonacci</h1>
    <div id="principal">
      <%
        long f1 = 0;
        long f2 = 1;
        long aux;
        int n = Integer.parseInt(request.getParameter("n"));

        switch (n) {
          case 1:
            out.print("0");
            break;

          case 2:
            out.print("0 1");
            break;

          default:
            out.print("0 1");
            while(n > 2) {
              aux = f1;
              f1 = f2;
              f2 = aux + f2;
              out.print(" " + f2);
              n--;
            }
        }
      %>
    </div>
```

```
  </body>
</html>
```

Fichero: index.jsp

```
<%--
14. Escribe un programa que muestre los n primeros términos de la serie de
    Fibonacci. El primer término de la serie de Fibonacci es 0, el segundo es 1
    y el resto se calcula sumando los dos anteriores, por lo que tendríamos que
    los términos son 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144... El número n
    se debe introducir por teclado.
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Relación 12 - Ejercicio 14</title>
    <link href="estilos.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <h1>Serie de Fibonacci</h1>
    <div id="principal">
    <form method="post" action="fibonacci.jsp">
      Esta aplicación muestra los <b>n</b> primeros números de la serie de Fibonacci.<br><br>
      Por favor, introduzca <b>n</b>:
      <input type="number" min="1" name="n">
      <input type="submit" value="Aceptar">
    </form>
    </div>
  </body>
</html>
```

Fichero: estilos.css

```
root {
  display: block;
}

body {
  font-family: Arial, sans-serif;
  background-color: #bcdeff;
}
```

```
p {
  text-align: center;
}

h1 {
  text-align: center;
  color: white;
  text-shadow: #242424 2px 2px 1px;
}

#principal {
  background-color: #abcdef;
  width: 600px;
  margin: auto;
  padding: 60px;
  box-shadow: #242424 4px 4px 4px;
  border-radius: 6px;
  text-align: justify;
}
```

Ejercicio 15

Fichero: index.jsp

```
<%--
15. Realiza una aplicación que genere 100 números aleatorios del 1 al 200. Los
    primos deberán aparecer en un color diferente al resto.
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Relación 12 - Ejercicio 15</title>
    <link href="estilos.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <div id="principal">
    <p>
      Esta aplicación genera 100 números aleatorios del 1 al 200. Los primos aparecen
      en color verde.
    </p>
    <p>
      <%
```

```
        int n;
        for (int i = 0; i < 100; i++) {
          n = (int)(Math.random() * 200) + 1;
          if (esPrimo(n)) {
            out.print("<span class=\"verde\">" + n + "</span> ");
          } else {
            out.print(n + " ");
          }
        }
      %>
    </p>
    <p>
      Pulsa la tecla <b>F5</b> para ejecutar de nuevo la aplicación.
    </p>
    </div>
  </body>
</html>

<!-- FUNCIONES -->
<%!
  static boolean esPrimo(int n) {
    for (int i = 2; i < n; i++) {
      if (n % i == 0) {
        return false;
      }
    }
    return true;
  }
%>
```

Fichero: estilos.css

```
root {
  display: block;
}

body {
  font-family: Arial, sans-serif;
  background-color: #bcdeff;
}

p {
  text-align: justify;
}

h1 {
```

```
  text-align: center;
  color: white;
  text-shadow: #242424 2px 2px 1px;
}

#principal {
  background-color: #abcdef;
  width: 600px;
  margin: auto;
  padding: 60px;
  box-shadow: #242424 4px 4px 4px;
  border-radius: 6px;
  text-align: justify;
}

.verde { color: green; }
```

Ejercicio 16

Fichero: index.jsp

```
<%--
16. Realiza una aplicación que muestre la tirada aleatoria de tres dados de
    póker. Utiliza imágenes de dados reales.
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Relación 12 - Ejercicio 16</title>
    <link href="estilos.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <div id="principal">
      <h1>Esta aplicación muestra la tirada aleatoria de tres dados de póker.</h1>
      <p>
        <%
          String[] cara = {"as.png", "j.png", "q.png", "k.png", "siete.png", "ocho.png"};

          for (int i = 0; i < 3; i++) {
            out.print("<img src=\"" + cara[(int)(Math.random()*6)] +"\">");
          }
        %>
```

```
      </p>
      <h1>Pulsa la tecla <b>F5</b> para ejecutar de nuevo la aplicación.</h1>
    </div>
  </body>
</html>
```

Fichero: estilos.css

```
root {
  display: block;
}

body {
  font-family: Arial, sans-serif;
  background-color: #bcdeff;
}

p {
  text-align: justify;
}

h1 {
  font-size: 24px;
  text-align: center;
  color: white;
  text-shadow: #242424 2px 2px 1px;
  padding: 20px;
}

#principal {
  background-image: url(texturaverde.jpg);
  width: 800px;
  margin: auto;
  margin-top: 20px;
  padding: 60px;
  box-shadow: #242424 4px 4px 4px;
  border-radius: 6px;
  text-align: justify;
}
```

Ejercicio 17

Fichero: coche.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Relación 12 - Ejercicio 17</title>
    <link href="estilos.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <h1>Aquí tiene su coche, enhorabuena.</h1>
    <%
      String tapiceria = request.getParameter("tapiceria");
      String moldura = request.getParameter("moldura");
      String imagen = tapiceria + moldura + ".jpg";
    %>
    <p><img src="<%= imagen %>"></p>
  </body>
</html>
```

Fichero: index.jsp

```
<%--
  17. Realiza un configurador del interior de un vehículo. El usuario puede
      elegir, mediante un formulario, el color de la tapicería – blanco, negro
      o berenjena - y el material de las molduras – madera o carbono. Se debe
      mostrar el interior del coche tal y como lo quiere el usuario.
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Relación 12 - Ejercicio 17</title>
    <link href="estilos.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <h1>Configurador de vehículo</h1>
    <form method="get" action="coche.jsp">
      Tapicería:
      <select name="tapiceria">
        <option value="negro" selected="selected">Cuero negro</option>
        <option value="berenjena">Color berenjena</option>
        <option value="blanco">Blanco marfil</option>
      </select>
      <br>
```

```
      <br>
      Tipo de moldura interior:
      <select name="moldura">
        <option value="carbono" selected="selected">Fibra de carbono</option>
        <option value="madera">Madera de nogal</option>
      </select>
      <br>
      <br>
      <input type="submit" value="Aceptar">
    </form>
  </body>
</html>
```

Fichero: estilos.css

```
root {
  display: block;
}

body {
  font-family: Arial, sans-serif;
  background-color: #bcdeff;
}

p {
  text-align: center;
}

h1 {
  text-align: center;
  color: white;
  text-shadow: #242424 2px 2px 1px;
}

h2 {
  text-align: center;
  padding-top: 32px;
  font-size: 18px;
}

form {
  background-color: #abcdef;
  width: 400px;
  margin: auto;
  padding: 60px;
  box-shadow: #242424 4px 4px 4px;
```

```
  border-radius: 6px;
  text-align: center;
}

table {
  background-color: #abcdef;
  margin: auto;
  padding: 20px;
  box-shadow: #242424 4px 4px 4px;
  text-align: right;
  border: 0px;
  border-collapse: collapse;
}

th {
  background-color: #aaaadd;
  color: white;
  padding: 4px;
  margin: 2px;
}

td {
  padding: 10px;
}
```

Ejercicio 18

Fichero: index.jsp

```
<%--
  18. Crea la aplicación "El Trile". Deben aparecer tres cubiletes por pantalla
      y el usuario deberá seleccionar uno de ellos. Para dicha selección se
      puede usar un formulario con radio-button, una lista desplegable, hacer
      clic en el cubilete o lo que resulte más fácil. Se levantarán los tres
      cubiletes y se verá si estaba o no la bolita dentro del que el usuario
      indicó, así mismo, se mostrará un mensaje diciendo "Lo siento, no has
      acertado" o "¡Enhorabuena!, has encontrado la bolita". La probabilidad de
      encontrar la bolita es de 1/3.
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

```
    <title>Relación 12 - Ejercicio 18</title>
    <link href="estilos.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <h1>El Trile</h1>
    <table>
      <tr>
        <td><a href="resultado.jsp?cubilete=0"><img src="cubilete_tapando_bolas.png"</a></td>
        <td><a href="resultado.jsp?cubilete=1"><img src="cubilete_tapando_bolas.png"</a></td>
        <td><a href="resultado.jsp?cubilete=2"><img src="cubilete_tapando_bolas.png"</a></td>
      </tr>
    </table>
    <p>Adivina dónde está la bolita.</p>
  </body>
</html>
```

Fichero: resultado.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Relación 12 - Ejercicio 18</title>
    <link href="estilos.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <h1>El Trile</h1>
    <%
      int bolita = (int)(Math.random() * 3);
      int cubilete = Integer.parseInt(request.getParameter("cubilete"));

      String imagen[] = {"cubilete_sin_bola.png", "cubilete_sin_bola.png", "cubilete_sin_bola.\
png"};
      imagen[bolita] = "cubilete_con_1bola.png";

      String mensaje;
      if (bolita == cubilete) {
        mensaje = "¡Enhorabuena! ¡Has acertado!";
      } else {
        mensaje = "Lo siento, has perdido.";
      }

    %>
    <table>
      <tr>
```

```
        <td><img src="<%= imagen[0] %>"></td>
        <td><img src="<%= imagen[1] %>"></td>
        <td><img src="<%= imagen[2] %>"></td>
      </tr>
    </table>
      <p><%= mensaje %></p>
      <p><a href="index.jsp"><button>Volver a jugar</button></a></p>

  </body>
</html>
```

Fichero: estilos.css

```
root {
  display: block;
}

body {
  font-family: Arial, sans-serif;
  background-color: #bcdeff;
}

p {
  text-align: center;
}

h1 {
  text-align: center;
  color: white;
  text-shadow: #242424 2px 2px 1px;
}

h2 {
  text-align: center;
  padding-top: 32px;
  font-size: 18px;
}

form {
  background-color: #abcdef;
  width: 400px;
  margin: auto;
  padding: 60px;
  box-shadow: #242424 4px 4px 4px;
  border-radius: 6px;
  text-align: center;
```

```
}

table {
  background-color: #abcdef;
  margin: auto;
  padding: 20px;
  box-shadow: #242424 4px 4px 4px;
  text-align: right;
  border: 0px;
  border-collapse: collapse;
}

th {
  background-color: #aaaadd;
  color: white;
  padding: 4px;
  margin: 2px;
}

td {
  padding: 10px;
}
```

Ejercicio 19

Fichero: index.jsp

```
<%--
  19. Crea el juego "Apuesta y gana". El usuario debe introducir inicialmente una cantidad de
      dinero. A continuación aparecerá por pantalla una imagen de forma aleatoria. Si sale una
      calavera, el usuario pierde todo su dinero y termina el juego. Si sale medio limón, el u\
suario
      pierde la mitad del dinero y puede seguir jugando con esa cantidad o puede dejar de juga\
r.
      Si sale el gato chino de la suerte, el usuario multiplica por dos su dinero y puede segu\
ir
      jugando con esa cantidad o puede dejar de jugar.
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Relación 12 - Ejercicio 19</title>
```

```
    <link href="css/estilos.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <h1>Apuesta y gana</h1>
    <div id="principal">
    <form method="post" action="juego.jsp">
      <p>Por favor, introduzca la cantidad que quiere apostar:</p>
      <input type="number" min="1" name="dinero">€<br>
      <input type="submit" value="Aceptar">
    </form>
    </div>
  </body>
</html>
```

Fichero: juego.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <link href="css/estilos.css" rel="stylesheet" type="text/css" />
  </head>
    <h1>Apuesta y gana</h1>
    <div id="principal">
      <%
        int dinero = Integer.parseInt(request.getParameter("dinero"));
        int jugada = (int)(Math.random() * 3);
        String[] imagen = {"gatochinosuerte.gif", "mediolimon.jpg", "calavera.png"};
        String[] mensaje = {"Ha doblado el dinero.", "Ha perdido la mitad.", "Lo siento, ha pe\
rdido."};

        if (jugada == 0) { // Gato chino de la suerte: se dobla el dinero
          dinero *= 2;
        } else if (jugada == 1) { // Medio limón: se pierde la mitad
          dinero /= 2;
        } else { // Calavera: el jugador pierde.
          dinero = 0;
        }
      %>

        <p>
          <img src="img/<%= imagen[jugada] %>"><br>
          <h2><%= mensaje[jugada] %></h2>
        </p>
```

```
      <%
        if ((jugada == 0) || (jugada == 1)) { // El jugador puede seguir jugando o plantarse
      %>
          <h2>Ahora tiene <%= dinero %> €</h2>
          <form method="post" action="juego.jsp">
            <input type="hidden" name="dinero" value="<%= dinero %>">
            <input type="submit" value="Sigo jugando">
          </form>

          <form method="post" action="fin.jsp">
            <input type="hidden" name="dinero" value="<%= dinero %>">
            <input type="submit" value="Me planto">
          </form>
      <%
        } else {
      %>
          <form method="post" action="index.jsp">
            <input type="submit" value="Volver a jugar">
          </form>
      <%
        } // if
      %>
    </div>
</html>
```

Fichero: fin.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <link href="css/estilos.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <h1>Apuesta y gana</h1>
    <div id="principal">
      <%
        int dinero = Integer.parseInt(request.getParameter("dinero"));
      %>
      <h2>Ha conseguido <%= dinero %> euros</h2>
      <form method="post" action="index.jsp">
        <input type="submit" value="Volver a jugar">
      </form>
    </div>
  </body>
```

```
</html>
```

Ejercicio 20

Fichero: index.jsp

```
<%--
  20. Crea una aplicación que dibuje un tablero de ajedrez mediante una tabla
      HTML generada con bucles usando JSP y que sitúe dentro del tablero un
      alfil y un caballo en posiciones aleatorias. Las dos figuras no pueden
      estar colocadas en la misma casilla. Las filas y las columnas del tablero
      deben estar etiquetadas correctamente.
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Relación 12 - Ejercicio 20</title>
    <link href="css/estilos.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <h1>Ejercicio 20</h1>
    <div id="principal">
    <table>
      <tr><td></td><td>a</td><td>b</td><td>c</td><td>d</td><td>e</td><td>f</td><td>g</td><td>h\
</td><td></td></tr>
      <%
      String color = "";
      String imagen = "";

      int filaAlfil;
      int columnaAlfil;
      int filaCaballo;
      int columnaCaballo;

      do {

        // Coordenadas del alfil
        filaAlfil = (int)(Math.random() * 8) + 1;
        columnaAlfil = (int)(Math.random() * 8);

        // Coordenadas del caballo
        filaCaballo = (int)(Math.random() * 8) + 1;
```

```
        columnaCaballo = (int)(Math.random() * 8);

      } while ((filaAlfil == filaCaballo) && (columnaAlfil == columnaCaballo));


      for (int fila = 8; fila > 0; fila--) {
        out.print("<tr><td>" + fila + "</td>");
        for (int columna = 0; columna < 8; columna++) {

          // Determina el color de la casilla
          if ((((fila % 2) + (columna % 2)) % 2) == 0) {
            color = "blanco";
          } else {
            color = "negro";
          }

          // Determina la imagen que se inserta en la casilla
          if ((fila == filaAlfil) && (columna == columnaAlfil)) {
            imagen = "alfilblanco.png";
          } else if ((fila == filaCaballo) && (columna == columnaCaballo)) {
            imagen = "caballonegro.png";
          } else {
            imagen = "transparente.png";
          }

          out.print("<td class=\""+ color + "\"><img src=\"img/" + imagen + "\"</td>");
        }
        out.print("<td>" + fila + "</td></tr>");
      }
      %>
      <tr><td></td><td>a</td><td>b</td><td>c</td><td>d</td><td>e</td><td>f</td><td>g</td><td>h\
</td><td></td></tr>
    </table>
    </div>
  </body>
</html>
```

Ejercicio 21

Fichero: index.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Máquina de helados</title>
    <style>
      .contenedor {
        display: flex;
        width: 100%;
        justify-content: center;
      }

      .sabor {
        text-align: center;
        padding: 1em;
      }

      .error {
        text-align: center;
        color: red;
      }
    </style>
  </head>
  <body>

    <h1 style="text-align: center;">Máquina de helados</h1>
    <p style="text-align: center;">Seleccione los porcentajes para preparar un helado a su gus\
to.</p>
    <form action="helado.jsp">

      <div class="contenedor">
        <div class="sabor">
          <img src="img/chocolate.jpg" width="200px"><br>
          Chocolate
          <input type="number" value=0 min="0" max="100" name="chocolate"> %
        </div>
        <div class="sabor">
          <img src="img/fresa.jpg" width="200px"><br>
          Fresa:
          <input type="number" value=0 min="0" max="100" name="fresa"> %
        </div>
        <div class="sabor">
          <img src="img/vainilla.jpg" width="200px"><br>
          Vainilla:
          <input type="number" value=0 min="0" max="100" name="vainilla"> %
```

```
        </div>
      </div>

      <div class="contenedor">
        <div>
          <input type="submit" value="Preparar helado">
        </div>
      </div>

    </form>

    <div class="error">
      <p>
      <%=
        session.getAttribute("error") == null ?
                "" : session.getAttribute("error")
      %>
      <% session.removeAttribute("error"); %>
      </p>
    </div>
  </body>
</html>
```

Fichero: helado.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Máquina de helados</title>
    <style>
      .contenedor {
        display: flex;
        width: 100%;
        justify-content: center;
      }
      .tarrina {
        width: 320px;
        border-width: 6px;
        border-radius: 4px;
        border-color: brown;
        border-style: solid;
        border-top-color: white;
      }
    </style>
```

```
  </head>
  <body>
    <%
      int chocolate = Integer.parseInt(request.getParameter("chocolate"));
      int fresa = Integer.parseInt(request.getParameter("fresa"));
      int vainilla = Integer.parseInt(request.getParameter("vainilla"));

      int vacio = 100 - (chocolate + fresa + vainilla);

      if (chocolate + fresa + vainilla > 100) {
        session.setAttribute("error", "La suma de porcentajes no pueden ser mayor que el 100%"\
);
        response.sendRedirect("index.jsp");
      } else {
    %>
    <h1 style="text-align: center;">Aquí tiene su tarrina de helado</h1>

    <div class="contenedor">
      <div class="tarrina" style="">

        <!-- Espacio vacío hasta completar el tamaño de la tarrina -->
        <div style="height: <%= vacio %>px"></div>

        <% if (chocolate > 0) { %>
        <!-- Chocolate -->
        <div style="height: <%= chocolate %>px; background-color: #57331b;">Chocolate <%= choc\
olate %>%</div>
        <% } %>

        <% if (fresa > 0) { %>
        <!-- Fresa -->
        <div style="height: <%= fresa %>px; background-color: #ecacd6;">Fresa <%= fresa %>%</d\
iv>
        <% } %>

        <% if (vainilla > 0) { %>
        <!-- Vainilla -->
        <div style="height: <%= vainilla %>px; background-color: #f7d88e;">Vainilla <%= vainil\
la %>%</div>
        <% } %>

      </div>
    </div>
    <%
      }
    %>
```

```
  </body>
</html>
```

Ejercicio 22

Fichero: index.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Menú</title>
  </head>
  <body>

    <%
      String[] imagenes = {"hamburguesa", "pasta", "pizza", "quinoa", "agua", "cerveza", "refr\
esco"};
    %>

    <h1 style="text-align: center;">Pide la comida más sana a Domicilio</h1>

    <form action="pedido.jsp">
      <div style="display: flex; width: 100%; justify-content: center;">
        <%
          for (int i = 0; i < 4; i++) {
        %>
        <div>
          <img src="imagenes/<%= imagenes[i]%>.png" width="200px"><br>
          <input type="number" name="comida<%= i%>" value="0" min="0">
        </div>
        <%
          }
        %>
      </div>

      <div style="display: flex; width: 100%; justify-content: center;">
        <%
          for (int i = 4; i < 7; i++) {
        %>
        <div>
          <img src="imagenes/<%= imagenes[i]%>.png" width="100px"><br>
          <input type="number" name="comida<%= i%>" value="0" min="0">
        </div>
```

```
        <%
          }
        %>
      </div>
      <br><br>
      <div style="display: flex; width: 100%; justify-content: center;">
        <button style="text-align: center" type="submit">Hacer pedido</button>
      </div>
    </form>
  </body>
</html>
```

Fichero: pedido.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bo\
otstrap.min.css" integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2\
MZw1T" crossorigin="anonymous">
        <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js" i\
ntegrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM" crossorigin\
="anonymous"></script>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Pedido</title>
    </head>
    <body>
      <%
        int[] cantidades = new int[7];
        double[] precios = { 6.20, 5.50, 4.90, 6.95, 1, 1.5, 1.4};
        String[] descripciones = {"Hamburguesa vegetariana", "Pasta al pesto", "Pizza caprese"\
, "Quinoa con verdura", "Agua", "Cerveza", "Refresco"};

        for (int i = 0; i < 7; i++) {
          cantidades[i] = Integer.parseInt(request.getParameter("comida" + i));
        }
      %>
        <div style="text-align: center; margin: 0 auto; padding: 10px">
            <h1>Aquí tiene su pedido</h1><hr>
            <div class="row">
                <div class="col">Comida/bebida</div>
                <div class="col">PVP</div>
                <div class="col">Cantidad</div>
                <div class="col">Subtotal</div>
            </div>
```

```
          <%
            for (int i = 0; i < 7; i++) {
              if (cantidades[i] > 0) {
          %>
              <div class="row">
                <div class="col"><%= descripciones[i] %></div>
                <div class="col"><%= precios[i] %></div>
                <div class="col"><%= cantidades[i] %></div>
                <div class="col"><%= cantidades[i] * precios[i] %></div>
              </div>
          <%
              } // if
            } // for
          %>

          <%
            double total = 0;

            for (int i = 0; i < 7; i++) {
              total += cantidades[i] * precios[i];
            }
          %>

          <div class="row">
            <div class="col">
              <b>Total: <%= String.format("%.2f", total) %> €</b>
            </div>
            <div class="col"></div>
            <div class="col"></div>
            <div class="col"></div>
          </div>

        </div>
    </body>
</html>
```

Ejercicio 23

Fichero: index.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Tres en raya</title>
    <style>
      td {
        border: lightgray solid 1px;
      }
    </style>
  </head>
  <body>
    <%
      String[][] n = new String[3][3];

      for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
          n[i][j] = "vacio";
        }
      }

      int circulos = 0;
      do {
        int i = (int) (Math.random() * 3);
        int j = (int) (Math.random() * 3);
        if (n[i][j].equals("vacio")) {
          n[i][j] = "circulo";
          circulos++;
        }
      } while (circulos < 3);

      int cruces = 0;
      do {
        int i = (int) (Math.random() * 3);
        int j = (int) (Math.random() * 3);
        if (n[i][j].equals("vacio")) {
          n[i][j] = "cruz";
          cruces++;
        }
      } while (cruces < 3);
    %>
    <table>
      <%
      for (int i = 0; i < 3; i++) {
      %>
```

```
      <tr>
        <%
        for (int j = 0; j < 3; j++) {
        %>
          <td><img src="img/<%= n[i][j] %>.png"></td>
        <%
        }
      }
      %>
    </table>
  </body>
</html>
```

Ejercicio 24

Fichero: index.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Resultados electorales</title>
    <style>
      .contenedor {
        display: flex;
        width: 100%;
        justify-content: center;
        flex-direction: column;
      }

      .partido, .boton {
        text-align: center;
        padding: 1em;
      }

      .error {
        text-align: center;
        color: red;
      }
    </style>
  </head>
  <body>

    <h1 style="text-align: center;">Resultados electorales</h1>
```

```
    <p style="text-align: center;">Introduzca el número de votos de los partidos políticos.</p>
    <form action="grafica.jsp">

      <div class="contenedor">
        <div class="partido">
          PP <input type="number">
        </div>
        <div class="partido">
          PSOE <input type="number">
        </div>

        <div class="partido">
          Ciudadanos <input type="number">
        </div>

        <div class="partido">
          Vox <input type="number">
        </div>

        <div class="partido">
          PACMA <input type="number">
        </div>

        <div class="partido">
          Podemos <input type="number">
        </div>

        <div class="boton">
          <input type="submit" value="Ver gráfica">
        </div>
      </div>

    </form>

  </body>
</html>
```

Fichero: grafica.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Resultados electorales</title>
    <style>
      .contenedor {
        display: flex;
        width: 100%;
        justify-content: center;
      }

      table {
        border: 2px solid gray;
        border-collapse: collapse;
      }

      td {
        border: 1px solid gray;
        text-align: center;
        padding: 2px;

      }
    </style>
    <script src="js/Chart.js"></script>
  </head>
  <body>
    <h1 style="text-align: center;">Resultados electorales</h1>

    <div class="contenedor">
      <canvas id="pie-chart" height="100%"></canvas>
    </div>

    <div class="contenedor">

      <table>
        <thead>
          <tr>
            <th></th>
            <th>PP</th>
            <th>PSOE</th>
            <th>Ciudadanos</th>
            <th>Vox</th>
            <th>PACMA</th>
            <th>Podemos</th>
```

```
          <th>TOTAL</th>
        </tr>

      </thead>
      <tbody>
        <tr>
          <th>Votos</th>
          <td>4356023</td>
          <td>7480755</td>
          <td>4136600</td>
          <td>2677173</td>
          <td>326045</td>
          <td>3732929</td>
          <td>22709525</td>
        </tr>
        <tr>
          <th>Porcentaje</th>
          <td>19,18</td>
          <td>32,94</td>
          <td>18,22</td>
          <td>11,79</td>
          <td>1,44</td>
          <td>16,44</td>
          <td>100</td>
        </tr>
      </tbody>
    </table>

  </div>

</div>

<script>
  new Chart(document.getElementById("pie-chart"), {
    type: 'pie',
    data: {
      labels: ["PP", "PSOE", "Ciudadanos", "Vox", "PACMA", "Podemos"],
      datasets: [{
          backgroundColor: ["#0055A7", "#FF2527", "#FE6139", "#66BC29", "#FEDB44", "#6A3468"\
],
          data: [4356023, 7480755, 4136600, 2677173, 326045, 3732929]
        }]
    }
  });
</script>
```

```
</body>
</html>
```

Ejercicio 25

Fichero: index.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Ejercicio 25</title>
  </head>
  <body>
    <%
      int dado1 = 0;
      int dado2 = 0;
      int dado3 = 0;

      do {
        dado1 = (int)(Math.random() * 6 + 1);
        dado2 = (int)(Math.random() * 6 + 1);
        dado3 = (int)(Math.random() * 6 + 1);
        %>
        <img src="img/<%= dado1 %>.png">
        <img src="img/<%= dado2 %>.png">
        <img src="img/<%= dado3 %>.png">
        <br>
        <%
      } while (!((dado1 == dado2) && (dado2 == dado3)));
    %>
  </body>
</html>
```

Acceso a bases de datos

Ejercicio 1

Fichero: acceso.jsp

```
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.Connection"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

    <!-- Materialize -->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/css\
/materialize.min.css">
    <link href="http://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
  </head>

  <body>
  <%
    Class.forName("com.mysql.jdbc.Driver");
    Connection conexion = DriverManager.getConnection("jdbc:mysql://localhost:3306/seguridad",\
"root", "root");
    Statement s = conexion.createStatement();

    request.setCharacterEncoding("UTF-8");

    String consulta = "SELECT COUNT(*) FROM acceso WHERE usuario='"
                      + request.getParameter("usuario")
                      + "' AND clave='"
                      + request.getParameter("clave")
                      + "'";

    ResultSet coincidencias = s.executeQuery(consulta);
    coincidencias.next();

    String icono;
    String mensaje;
    String enlace;

    if (coincidencias.getInt(1) == 0) {
```

```
      icono = "<i class=\"material-icons red-text large\">lock</i>"; // candado cerrado
      mensaje = "<p>Lo siento, acceso denegado.</p>";
      enlace = "index.jsp";
      //out.print("<script type=\"text/javascript\">alert(\"Lo siento, acceso denegado\");</sc\
ript>");
      //out.print("<script>document.location = \"index.jsp\"</script>");
    } else {
      icono = "<i class=\"material-icons teal-text large\">lock_open</i>";
      mensaje = "<p>Acceso permitido a la aplicación.</p>"; // candado abierto
      enlace = "indexp.jsp";
      //out.print("<script type=\"text/javascript\">alert(\"Acceso permitido a la aplicación\"\
);</script>");
      //out.print("<script>document.location = \"indexp.jsp\"</script>");
    };
    %>
    <div class="container">
      <div class="row"></div>
      <div class="row">
        <div class="row col m3"></div>
        <div class="col m6 card-panel grey lighten-5 center">
          <h5 class="center">Control de acceso</h5>
          <%=icono %>
          <%=mensaje %>
          <p class="center">
            <a href="<%=enlace %>" class="btn waves-effect waves-light center">
              Aceptar
              <i class="material-icons">check_circle</i>
            </a>
          </p>
          <br>
        </div>
      </div>
    </div>

    <script type="text/javascript" src="https://code.jquery.com/jquery-2.1.1.min.js"></script>
    <!-- Materialize -->
    <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/js/materialize.min.\
js"></script>

  </body>
</html>
```

Fichero: indexp.jsp

```
<%--
9. Realiza una aplicación que pinte una pirámide. La altura se pedirá en un
formulario. La pirámide estará hecha de bolitas, ladrillos o cualquier otra
imagen.
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

    <!-- Materialize -->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/css\
/materialize.min.css">
    <link href="http://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
  </head>
  <body>
    <div class="container">
      <div class="row"></div>
      <div class="row">
        <div class="row col m4"></div>
        <div class="col m4 card-panel grey lighten-5">
          <h5 class="center">Pinta una pirámide</h5>
          <form method="post" action="piramide.jsp">
            <div class="input-field">
              <input type="number" min="1" max="10" name="altura" id="altura" required>
              <label for="altura">Altura</label>
            </div>
            <p class="center"><button class="btn waves-effect waves-light center" type="submit\
" name="aceptar">
              Aceptar
              <i class="material-icons">check_circle</i>
            </button></p>
            <br>
          </form>
        </div>
      </div>
    </div>

    <script type="text/javascript" src="https://code.jquery.com/jquery-2.1.1.min.js"></script>
    <!-- Materialize -->
    <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/js/materialize.min.\
js"></script>

  </body><!--
```

```
    <body>

        <form  class="form-4"method="get" action="piramide.jsp">
      <h1>Pinta una pirámide</h1>
      <p>
        <input type="text" name="altura" placeholder="Altura" required></br></br>
            </p>
      <p>
        <input type="submit" value="Aceptar">
      </p>
        </form>
    </body>-->
</html>
```

Fichero: piramide.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

    <!-- Materialize -->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/css\
/materialize.min.css">
    <link href="http://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
  </head>
    <body>
      <div class="container center">
        <%
          int alturaTotal = Integer.parseInt(request.getParameter("altura"));
          int altura = 1;
          int i = 0;

          while (altura <= alturaTotal) {
            // pinta la línea
            for (i = 1; i < altura*2; i++) {
              out.print("<img src=\"gatoloco.png\" width=\"40\">");
            }
            out.println("</br>");
            altura++;
          } // while
        %>
      </div>

      <script type="text/javascript" src="https://code.jquery.com/jquery-2.1.1.min.js"></scrip\
```

```
t>
      <!-- Materialize -->
      <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/js/materialize.mi\
n.js"></script>
    </body>
</html>
```

Fichero: index.jsp

```
<%--
  1. Establece un control de acceso mediante nombre de usuario y contraseña para cualquiera
  de los programas de la relación anterior. No se nos dejará ejecutar la aplicación hasta que
  iniciemos sesión con un nombre de usuario y contraseña correctos. Los nombres de usuario
  y contraseñas deben estar almacenados en una base de datos.
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

    <!-- Materialize -->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/css\
/materialize.min.css">
    <link href="http://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
  </head>
  <body>
    <div class="container">
      <div class="row"></div>
      <div class="row">
        <div class="row col m3"></div>
        <div class="col m6 card-panel grey lighten-5">
          <h5 class="center">Control de acceso</h5>
          <form method="post" action="acceso.jsp">
            <div class="input-field">
              <i class="material-icons prefix">account_circle</i>
              <input type="text" name="usuario" id="usuario" required>
              <label for="usuario">Usuario</label>
            </div>
            <div class="input-field">
              <i class="material-icons prefix">lock</i>
              <input type="password" name="clave" id="clave" required>
              <label for="clave">Contraseña</label>
            </div>
            <p class="center"><button class="btn waves-effect waves-light center" type="submit\
```

```
" name="aceptar">
              Aceptar
              <i class="material-icons">check_circle</i>
            </button></p>
            <br>
          </form>
        </div>
      </div>
    </div>

      <script type="text/javascript" src="https://code.jquery.com/jquery-2.1.1.min.js"></scrip\
t>
      <!-- Materialize -->
      <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/js/materialize.mi\
n.js"></script>
  </body>
</html>
```

Ejercicio 2

Fichero: altaUsuario.jsp

```
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.Connection"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <!-- Materialize -->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/css\
/materialize.min.css">
    <link href="http://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
    <title>Gestión de usuarios</title>
  </head>
  <body>
    <%
      Class.forName("com.mysql.jdbc.Driver");
      Connection conexion = DriverManager.getConnection("jdbc:mysql://localhost:3306/seguridad\
","root", "root");
      Statement s = conexion.createStatement();

      request.setCharacterEncoding("UTF-8");
```

```
      // Comprueba si el usuario ya existe
      String consulta = "SELECT COUNT(*) FROM acceso WHERE usuario='"
                      + request.getParameter("usuario")
                      + "'";

      ResultSet coincidencias = s.executeQuery(consulta);
      coincidencias.next();

      if (coincidencias.getInt(1) != 0) {
        out.print("<script type=\"text/javascript\">alert(\"Lo siento, el usuario " + request.\
getParameter("usuario") + " ya existe\");</script>");
        //out.print("<script>document.gestion_usuarios.jsp = \"index.jsp\"</script>");
      } else {
        // Inserta los datos en la base de datos
        String insercion = "INSERT INTO acceso VALUES ('"
                          + request.getParameter("usuario")
                          + "', '" + request.getParameter("clave") + "')";
        out.print(insercion);
        s.execute(insercion);
        conexion.close();
      }
    %>
    <script type="text/javascript" src="https://code.jquery.com/jquery-2.1.1.min.js"></script>
    <!-- Materialize -->
    <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/js/materialize.min.\
js"></script>

    <!--Vuelve a la gestión de usuarios-->
    <script>document.location = "gestionUsuarios.jsp"</script>
  </body>
</html>
```

Fichero: acceso.jsp

```
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.Connection"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <!-- Materialize -->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/css\
```

```
/materialize.min.css">
    <link href="http://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
  </head>

  <body>
  <%
    Class.forName("com.mysql.jdbc.Driver");
    Connection conexion = DriverManager.getConnection("jdbc:mysql://localhost:3306/seguridad",\
"root", "root");
    Statement s = conexion.createStatement();

    request.setCharacterEncoding("UTF-8");

    String consulta = "SELECT COUNT(*) FROM acceso WHERE usuario='"
                    + request.getParameter("usuario")
                    + "' AND clave='"
                    + request.getParameter("clave")
                    + "'";

    ResultSet coincidencias = s.executeQuery(consulta);
    coincidencias.next();

    String icono;
    String mensaje;
    String enlace;

    if (coincidencias.getInt(1) == 0) {
      icono = "<i class=\"material-icons red-text large\">lock</i>"; // candado cerrado
      mensaje = "<p>Lo siento, acceso denegado.</p>";
      enlace = "index.jsp";
    } else if (request.getParameter("usuario").equals("admin")) {
      icono = "<i class=\"material-icons large\">vpn_key</i>";
      mensaje = "<p>Tiene acceso al área de gestión de usuarios.</p>"; // candado abierto
      enlace = "gestionUsuarios.jsp";
    } else {
      icono = "<i class=\"material-icons teal-text large\">lock_open</i>";
      mensaje = "<p>Acceso permitido a la aplicación.</p>"; // candado abierto
      enlace = "indexp.jsp";
    };
    %>
    <div class="container">
      <div class="row"></div>
      <div class="row">
        <div class="row col m3"></div>
        <div class="col m6 card-panel grey lighten-5 center">
          <h5 class="center">Control de acceso</h5>
```

```
          <%=icono %>
          <%=mensaje %>
          <p class="center">
            <a href="<%=enlace %>" class="btn waves-effect waves-light center">
              Aceptar
              <i class="material-icons">check_circle</i>
            </a>
          </p>
          <br>
        </div>
      </div>
    </div>
    <script type="text/javascript" src="https://code.jquery.com/jquery-2.1.1.min.js"></script>
    <!-- Materialize -->
    <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/js/materialize.min.\
js"></script>
  </body>
</html>
```

Fichero: indexp.jsp

```
<%--
9. Realiza una aplicación que pinte una pirámide. La altura se pedirá en un
formulario. La pirámide estará hecha de bolitas, ladrillos o cualquier otra
imagen.
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <!-- Materialize -->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/css\
/materialize.min.css">
    <link href="http://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
  </head>
  <body>
    <div class="container">
      <div class="row"></div>
      <div class="row">
        <div class="row col m4"></div>
        <div class="col m4 card-panel grey lighten-5">
          <h5 class="center">Pinta una pirámide</h5>
          <form method="post" action="piramide.jsp">
            <div class="input-field">
```

```
              <input type="number" min="1" max="10" name="altura" id="altura" required>
              <label for="altura">Altura</label>
            </div>
            <p class="center"><button class="btn waves-effect waves-light center" type="submit\
" name="aceptar">
              Aceptar
              <i class="material-icons">check_circle</i>
            </button></p>
            <br>
          </form>
        </div>
      </div>
    </div>

    <script type="text/javascript" src="https://code.jquery.com/jquery-2.1.1.min.js"></script>
    <!-- Materialize -->
    <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/js/materialize.min.\
js"></script>
  </body>
</html>
```

Fichero: piramide.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <!-- Materialize -->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/css\
/materialize.min.css">
    <link href="http://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
  </head>
    <body>
      <div class="container center">
        <%
          int alturaTotal = Integer.parseInt(request.getParameter("altura"));
          int altura = 1;
          int i = 0;

          while (altura <= alturaTotal) {
            // pinta la línea
            for (i = 1; i < altura*2; i++) {
              out.print("<img src=\"gatoloco.png\" width=\"40\">");
            }
            out.println("</br>");
```

```
          altura++;
        } // while
      %>
    </div>
    <script type="text/javascript" src="https://code.jquery.com/jquery-2.1.1.min.js"></scrip\
t>
    <!-- Materialize -->
    <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/js/materialize.mi\
n.js"></script>
  </body>
</html>
```

Fichero: index.jsp

```
<%--
2. Mejora el programa anterior de tal forma que se puedan dar de alta nuevos
   usuarios para acceder a la aplicación. Si se introduce un nombre de usuario
   que no sea el administrador (admin) y una contraseña correcta, la aplicación
   funcionará exactamente igual que el ejercicio anterior. Si se introduce el
   usuario "admin" y la contraseña correcta, la aplicación entra en la gestión
   de usuarios donde se podrán dar de alta nuevos usuarios indicando nombre de
   usuario y contraseña. No puede haber dos nombres de usuario iguales aunque sí
   puede haber claves repetidas.
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <!-- Materialize -->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/css\
/materialize.min.css">
    <link href="http://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
  </head>
  <body>
    <div class="container">
      <div class="row"></div>
      <div class="row">
        <div class="row col m3"></div>
        <div class="col m6 card-panel grey lighten-5">
          <h5 class="center">Control de acceso</h5>
          <form method="post" action="acceso.jsp">
            <div class="input-field">
              <i class="material-icons prefix">account_circle</i>
              <input type="text" name="usuario" id="usuario" required>
```

```
          <label for="usuario">Usuario</label>
        </div>
        <div class="input-field">
          <i class="material-icons prefix">lock</i>
          <input type="password" name="clave" id="clave" required>
          <label for="clave">Contraseña</label>
        </div>
        <p class="center"><button class="btn waves-effect waves-light center" type="submit\
" name="aceptar">
          Aceptar
          <i class="material-icons">check_circle</i>
        </button></p>
        <br>
      </form>
    </div>
  </div>
</div>

<script type="text/javascript" src="https://code.jquery.com/jquery-2.1.1.min.js"></script>
<!-- Materialize -->
<script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/js/materialize.min.\
js"></script>
  </body>
</html>
```

Fichero: gestionUsuarios.jsp

```
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.Connection"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <!-- Materialize -->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/css\
/materialize.min.css">
    <link href="http://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
    <title>Gestión de usuarios</title>
  </head>

  <body>
    <%
      Class.forName("com.mysql.jdbc.Driver");
```

```
      Connection conexion = DriverManager.getConnection("jdbc:mysql://localhost:3306/seguridad\
","root", "root");
      Statement s = conexion.createStatement();

      request.setCharacterEncoding("UTF-8");

      ResultSet listado = s.executeQuery ("SELECT * FROM acceso");
    %>

    <div class="container">
      <div class="row"></div>
      <div class="row">
        <div class="col l2"> </div>
        <div class="col l8">
      <div class="card">
        <h5 class="center">Gestión de usuarios</h5>
        <table class="bordered striped centered responsive-table">
          <thead><th>Usuario</th><th>Contraseña</th></thead>
          <%
            while (listado.next()) {
              out.println("<tr><td>");
              out.println(listado.getString("usuario") + "</td>");
              out.println("<td>" + listado.getString("clave"));
              out.println("</td></tr>");
            }
          %>
        </table>
      </div>

      <div class="card">
        <table class="hoverable centered responsive-table">
          <form method="post" action="altaUsuario.jsp">
          <tr>
            <td>
              <div class="input-field">
                <input type="text" name="usuario" id="usuario" required>
                <label for="usuario">Usuario</label>
              </div>
            </td>
            <td>
              <div class="input-field">
                <input type="password" name="clave" id="clave" required>
                <label for="clave">Contraseña</label>
              </div>
            </td>
          </tr>
```

```
          <tr>
            <td colspan="2">
              <p class="center">
                <button class="btn waves-effect waves-light center" type="submit" name="acepta\
r">
                  Añadir usuario
                  <i class="material-icons">check_circle</i>
                </button>
              </p>
            </td>
          </tr>
          </form>
        </table>
      </div>
      </div>
      </div>
    </div>

    <%
      conexion.close();
    %>
    <script type="text/javascript" src="https://code.jquery.com/jquery-2.1.1.min.js"></script>
    <!-- Materialize -->
    <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/js/materialize.min.\
js"></script>
    </body>
</html>
```

Ejercicio 3

Fichero: altaUsuario.jsp

```
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.Connection"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <!-- Materialize -->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/css\
/materialize.min.css">
    <link href="http://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
```

```
    <title>Gestión de usuarios</title>
  </head>
  <body>
    <%
      Class.forName("com.mysql.jdbc.Driver");
      Connection conexion = DriverManager.getConnection("jdbc:mysql://localhost:3306/seguridad\
3","root", "root");
      Statement s = conexion.createStatement();

      request.setCharacterEncoding("UTF-8");

      // Comprueba si el usuario ya existe
      String consulta = "SELECT COUNT(*) FROM acceso WHERE usuario='"
                      + request.getParameter("usuario")
                      + "'";

      ResultSet coincidencias = s.executeQuery(consulta);
      coincidencias.next();

      if (coincidencias.getInt(1) != 0) {
        out.print("<script type=\"text/javascript\">alert(\"Lo siento, el usuario " + request.\
getParameter("usuario") + " ya existe\");</script>");
        //out.print("<script>document.gestion_usuarios.jsp = \"index.jsp\"</script>");
      } else {
        // Inserta los datos en la base de datos
        String insercion = "INSERT INTO acceso VALUES ('"
                         + request.getParameter("usuario")
                         + "', '" + request.getParameter("clave") + "')";
        out.print(insercion);
        s.execute(insercion);
        conexion.close();
      }
    %>
    <script type="text/javascript" src="https://code.jquery.com/jquery-2.1.1.min.js"></script>
    <!-- Materialize -->
    <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/js/materialize.min.\
js"></script>

    <!--Vuelve a la gestión de usuarios-->
    <script>document.location = "gestionUsuarios.jsp"</script>
  </body>
</html>
```

Fichero: acceso.jsp

```
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.Connection"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <!-- Materialize -->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/css\
/materialize.min.css">
    <link href="http://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
  </head>

  <body>
  <%
    Class.forName("com.mysql.jdbc.Driver");
    Connection conexion = DriverManager.getConnection("jdbc:mysql://localhost:3306/seguridad3"\
,"root", "root");
    Statement s = conexion.createStatement();

    request.setCharacterEncoding("UTF-8");

    String consulta = "SELECT COUNT(*) FROM acceso WHERE usuario='"
                    + request.getParameter("usuario")
                    + "' AND clave='"
                    + request.getParameter("clave")
                    + "'";

    ResultSet coincidencias = s.executeQuery(consulta);
    coincidencias.next();

    String icono;
    String mensaje;
    String enlace;

    if (coincidencias.getInt(1) == 0) {
      icono = "<i class=\"material-icons red-text large\">lock</i>"; // candado cerrado
      mensaje = "<p>Lo siento, acceso denegado.</p>";
      enlace = "index.jsp";
    } else if (request.getParameter("usuario").equals("admin")) {
      icono = "<i class=\"material-icons teal-text large\">supervisor_account</i>";
      mensaje = "<p>Tiene acceso al área de gestión de usuarios.</p>"; // candado abierto
      enlace = "gestionUsuarios.jsp";
    } else {
```

```
      icono = "<i class=\"material-icons teal-text large\">lock_open</i>";
      mensaje = "<p>Acceso permitido a sus aplicaciones.</p>"; // candado abierto
      enlace = "indexAplicaciones.jsp?usuario=" + request.getParameter("usuario");
    };
    %>
    <div class="container">
      <div class="row"></div>
      <div class="row">
        <div class="row col m3"></div>
        <div class="col m6 card-panel grey lighten-5 center">
          <h5 class="center">Control de acceso</h5>
          <%=icono %>
          <%=mensaje %>
          <p class="center">
            <a href="<%=enlace %>" class="btn waves-effect waves-light center">
              Aceptar
              <i class="material-icons">check_circle</i>
            </a>
          </p>
          <br>
        </div>
      </div>
    </div>
    <script type="text/javascript" src="https://code.jquery.com/jquery-2.1.1.min.js"></script>
    <!-- Materialize -->
    <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/js/materialize.min.\
js"></script>
  </body>
</html>
```

Fichero: indexAplicaciones.jsp

```
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.Connection"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <!-- Materialize -->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/css\
/materialize.min.css">
    <link href="http://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.4.0/css/font-a\
```

```
wesome.min.css">
  </head>

  <body>
    <%
      Class.forName("com.mysql.jdbc.Driver");
      Connection conexion = DriverManager.getConnection("jdbc:mysql://localhost:3306/seguridad\
3","root", "root");
      Statement s = conexion.createStatement();

      request.setCharacterEncoding("UTF-8");

      String consulta = "SELECT ejercicio FROM permiso WHERE usuario='"
                      + request.getParameter("usuario") + "'";
      ResultSet listado = s.executeQuery(consulta);
    %>
    <div class="container center">
      <div class="row"></div>
      <div class="row">
        <div class="col s2"> </div>
        <div class="col s8">
          <span>Como usuario <b><%=request.getParameter("usuario") %></b> puede ejecutar los s\
iguientes ejercicios:</span>
          <div class="collection">
            <%
              while (listado.next()) {
            %>
            <a href="EjerciciosJsp/S12Ejercicio<%=listado.getInt("ejercicio") %>/index.jsp" cl\
ass="collection-item">Ejercicio <%=listado.getInt("ejercicio") %></a>
            <%
              }
            %>
          </div>
          <p class="center">
            <a href="index.jsp" class="btn waves-effect waves-light center">
              <i class="fa fa-home"></i>
              Página principal
            </a>
          </p>
        </div>
      </div>
    </div>
    <script type="text/javascript" src="https://code.jquery.com/jquery-2.1.1.min.js"></script>
    <!-- Materialize -->
    <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/js/materialize.min.\
js"></script>
```

```
  </body>
</html>
```

Fichero: grabaPermisos.jsp

```
<%@page import="java.util.Vector"%>
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.Connection"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  </head>
  <body>
    <%
      Class.forName("com.mysql.jdbc.Driver");
      Connection conexion = DriverManager.getConnection("jdbc:mysql://localhost:3306/seguridad\
3","root", "root");
      Statement s = conexion.createStatement();

      request.setCharacterEncoding("UTF-8");

      s.execute("DELETE FROM permiso WHERE usuario='" + request.getParameter("usuario") + "'");

      for(int i = 1; i < 21; i++) {
        if(request.getParameter("ejer" + i) != null) {
          s.execute("INSERT INTO permiso VALUES ('" + request.getParameter("usuario") + "', " \
+ i + ")");
        }
      }
      conexion.close();
      %>
      <script>document.location = "gestionUsuarios.jsp"</script>
    </body>
</html>
```

Fichero: index.jsp

```
<%--
  3. Amplía el programa anterior para que se pueda asignar o quitar permiso para
     ejecutar las aplicaciones de la relación anterior a los distintos usuarios.
     Por ejemplo, que se pueda especificar que el usuario "jaimito" pueda
     ejecutar los ejercicios 2, 3 y 5. Para ello, en la base de datos deberá
     existir una tabla con las parejas (usuario, nº ejercicio). Por ejemplo, si
     el usuario “jaimito” tiene acceso a los ejercicios 2, 3 y 5; en la tabla
     correspondiente estarán las parejas (jaimito, 2), (jaimito, 3) y (jaimito, 5).
     Lo ideal es que la asignación de permisos se haga mediante el marcado de
     múltiples "checkbox".
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <!-- Materialize -->
    <link href="http://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/css\
/materialize.min.css">
  </head>
  <body>
    <div class="container">
      <div class="row"></div>
      <div class="row">
        <div class="row col m3"></div>
        <div class="col m6 card-panel grey lighten-5">
          <h5 class="center">Control de acceso</h5>
          <form method="post" action="acceso.jsp">
            <div class="input-field">
              <i class="material-icons prefix">account_circle</i>
              <input type="text" name="usuario" id="usuario" required>
              <label for="usuario">Usuario</label>
            </div>
            <div class="input-field">
              <i class="material-icons prefix">lock</i>
              <input type="password" name="clave" id="clave" required>
              <label for="clave">Contraseña</label>
            </div>
            <p class="center"><button class="btn waves-effect waves-light center" type="submit\
" name="aceptar">
              Aceptar
              <i class="material-icons">check_circle</i>
            </button></p>
            <br>
```

```
          </form>
        </div>
      </div>
    </div>

    <script type="text/javascript" src="https://code.jquery.com/jquery-2.1.1.min.js"></script>
    <!-- Materialize -->
    <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/js/materialize.min.\
js"></script>
  </body>
</html>
```

Fichero: gestionUsuarios.jsp

```
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.Connection"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <!-- Materialize -->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/css\
/materialize.min.css">
    <link href="http://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
    <title>Gestión de usuarios</title>
  </head>

  <body>
    <%
      Class.forName("com.mysql.jdbc.Driver");
      Connection conexion = DriverManager.getConnection("jdbc:mysql://localhost:3306/seguridad\
3","root", "root");
      Statement s = conexion.createStatement();

      request.setCharacterEncoding("UTF-8");

      ResultSet listado = s.executeQuery ("SELECT * FROM acceso");
    %>

    <div class="container">
      <div class="row"></div>
      <div class="row">
        <div class="col s1"> </div>
```

```
        <div class="col s10">
          <div class="card">
            <h5 class="center">Gestión de usuarios</h5>
            <table class="bordered striped centered responsive-table">
              <thead><th>Usuario</th><th>Contraseña</th><th>Permisos</th></thead>
              <%
                while (listado.next()) {
                  out.println("<tr><td>");
                  out.println(listado.getString("usuario") + "</td>");
                  out.println("<td>" + listado.getString("clave"));
                  %>
                  </td>
                  <td>
                    <form method="post" action="editaPermisos.jsp">
                      <input type="hidden" name="usuario" value="<%=listado.getString("usuario\
") %>">

                      <button class="btn waves-effect waves-light center blue" type="submit" n\
ame="editar">
                        Editar
                      </button>
                    </form>
                  </td></tr>
                  <%
                }
              %>

              <form method="post" action="altaUsuario.jsp">
                <tr>
                  <td>
                    <div class="input-field">
                      <input type="text" name="usuario" id="usuario" required>
                      <label for="usuario">Usuario</label>
                    </div>
                  </td>
                  <td>
                    <div class="input-field">
                      <input type="password" name="clave" id="clave" required>
                      <label for="clave">Contraseña</label>
                    </div>
                  </td>
                  <td>
                    <button class="btn waves-effect waves-light center" type="submit" name="ac\
eptar">
                      Añadir usuario
                      <i class="material-icons">check_circle</i>
                    </button>
```

```
                </td>
              </tr>
            </form>
          </table>
        </div>
      </div>
    </div>
  </div>

  <%
    conexion.close();
  %>
  <script type="text/javascript" src="https://code.jquery.com/jquery-2.1.1.min.js"></script>
  <!-- Materialize -->
  <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/js/materialize.min.\
js"></script>
  </body>
</html>
```

Fichero: editaPermisos.jsp

```
<%@page import="java.util.ArrayList"%>
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.Connection"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <!-- Materialize -->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/css\
/materialize.min.css">
    <link href="http://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
    <title>Permisos</title>
  </head>

  <body>
    <%
      Class.forName("com.mysql.jdbc.Driver");
      Connection conexion = DriverManager.getConnection("jdbc:mysql://localhost:3306/seguridad\
3","root", "root");
      Statement s = conexion.createStatement();

      request.setCharacterEncoding("UTF-8");
      String consulta = "SELECT ejercicio FROM permiso WHERE usuario='"
```

```
                        + request.getParameter("usuario") + "'";

      ResultSet listado = s.executeQuery(consulta);

      ArrayList<Integer> a = new ArrayList<Integer>();

      while (listado.next()) {
        a.add(listado.getInt("ejercicio"));
      }

      conexion.close();
    %>

    <div class="container">
      <div class="row"></div>
      <div class="card">
        <h5 class="center">Permisos</h5>
        <table class="bordered centered responsive-table">
            <thead><th>Usuario</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th\
><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</\
th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th></thead>
            <tr><td><%=request.getParameter("usuario") %></td>
            <form method="post" action="grabaPermisos.jsp">
            <%
              for(int i = 1; i < 21; i++) {
                %>
                <div class="input-field">
                <td>
                <input type="checkbox" name="ejer<%=i %>" id="ejer<%=i %>"
                <%
                if (a.contains(i)) {
                  out.print(" checked=\"checked\"");
                }
                %>
                ><label for="ejer<%=i %>"></label>
                </td>
                </div>
              <%
              }
            %>

            <td>
              </div>
              <input type="hidden" name="usuario" value="<%=request.getParameter("usuario") %>\
">
              <button class="btn waves-effect waves-light center" type="submit" name="aceptar">
```

```
                <i class="material-icons">check_circle</i>
              </button>
            </td></tr>
        </table>
      </div>
    </div>
    <script type="text/javascript" src="https://code.jquery.com/jquery-2.1.1.min.js"></script>
    <!-- Materialize -->
    <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/js/materialize.min.\
js"></script>
    <script>
      $(document).ready(function() {
        $('select').material_select();
      });
    </script>
    </body>
</html>
```

Ejercicio 4

Fichero: grabaCliente.jsp

```
<%@page import="java.util.Vector"%>
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.Connection"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  </head>
  <body>
    <%
      Class.forName("com.mysql.jdbc.Driver");
      Connection conexion = DriverManager.getConnection("jdbc:mysql://localhost:3306/banco","r\
oot", "root");
      Statement s = conexion.createStatement();

      request.setCharacterEncoding("UTF-8");

      s.execute("UPDATE cliente SET nombre='" + request.getParameter("nombre") + "', direccion\
='" + request.getParameter("direccion") + "', telefono='" + request.getParameter("telefono") +\
 "', nacimiento='" + request.getParameter("nacimiento") + "' WHERE clienteid=" + request.getPa\
```

```
rameter("clienteid"));

      conexion.close();
    %>
    <script>document.location = "index.jsp"</script>
  </body>
</html>
```

Fichero: borraCliente.jsp

```
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.Connection"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  </head>
  <body>
    <%
      Class.forName("com.mysql.jdbc.Driver");
      Connection conexion = DriverManager.getConnection("jdbc:mysql://localhost:3306/banco", "\
root", "root");
      Statement s = conexion.createStatement();

      s.execute("DELETE FROM cliente WHERE clienteid=" + request.getParameter("clienteid"));
    %>
    <script>document.location = "index.jsp"</script>
  </body>
</html>
```

Fichero: index.jsp

```
<%--
  4. Crea una aplicación web que permita hacer listado, alta, baja y
     modificación sobre la tabla cliente de la base de datos banco. Un cliente
     tiene su identificador, nombre completo, dirección, teléfono y fecha de
     nacimiento.
--%>

<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.DriverManager"%>
```

```
<%@page import="java.sql.Connection"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <!-- Materialize -->
    <link href="http://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/css\
/materialize.min.css">
    <title>Gestibank</title>
  </head>

  <body>
    <%
      Class.forName("com.mysql.jdbc.Driver");
      Connection conexion = DriverManager.getConnection("jdbc:mysql://localhost:3306/banco","r\
oot", "root");
      Statement s = conexion.createStatement();

      request.setCharacterEncoding("UTF-8");

      ResultSet listado = s.executeQuery ("SELECT * FROM cliente");
    %>

    <div class="container">
      <div class="row"></div>
      <div class="row s12 m12 l12">
          <div class="card grey lighten-5">
            <h3 class="center">G e s t i b a n k</h3>
            <table class="bordered centered responsive-table">
              <thead><th>Código</th><th>Nombre</th><th>Dirección</th><th>Teléfono</th><th>Fech\
a de nacimiento</th><th></th><th></th></thead>
              <%
                while (listado.next()) {
              %>
                  <tr>
                    <td><%=listado.getString("clienteid") %></td>
                    <td><%=listado.getString("nombre") %></td>
                    <td><%=listado.getString("direccion") %></td>
                    <td><%=listado.getString("telefono") %></td>
                    <td><%=listado.getString("nacimiento") %></td>
                    <form method="post" action="modificaCliente.jsp">
                      <input type="hidden" name="clienteid" value="<%=listado.getString("clien\
teid") %>">
                      <input type="hidden" name="nombre" value="<%=listado.getString("nombre")\
```

```
 %>">
                      <input type="hidden" name="direccion" value="<%=listado.getString("direc\
cion") %>">
                      <input type="hidden" name="telefono" value="<%=listado.getString("telefo\
no") %>">
                      <input type="hidden" name="nacimiento" value="<%=listado.getString("naci\
miento") %>">
                      <td>
                      <button class="btn waves-effect waves-light center blue" type="submit" n\
ame="editar">
                        Editar
                      </button>
                      </td>
                    </form>
                    <form method="post" action="borraCliente.jsp">
                        <input type="hidden" name="clienteid" value="<%=listado.getString("cli\
enteid") %>">
                      <td>
                      <button class="btn waves-effect waves-light center red" type="submit" na\
me="borrar">
                        Borrar
                      </button>
                      </td>
                    </form>
                  </tr>
              <%
                }
              %>
              <form method="post" action="altaCliente.jsp">
                <tr>
                  <td>
                    <div class="input-field">
                      <input type="number" name="clienteid" id="clienteid" required>
                      <label for="clienteid">código</label>
                    </div>
                  </td>
                  <td>
                    <div class="input-field">
                      <input type="text" name="nombre" id="nombre">
                      <label for="nombre">nombre</label>
                    </div>
                  </td>
                  <td>
                    <div class="input-field">
                      <input type="text" name="direccion" id="direccion">
                      <label for="direccion">dirección</label>
```

```
                    </div>
                  </td>
                  <td>
                    <div class="input-field">
                      <input type="tel" name="telefono" id="telefono">
                      <label for="telefono">teléfono</label>
                    </div>
                  </td>
                  <td>
                    <div class="input-field">
                      <input type="date" class="datepicker" name="nacimiento" id="nacimiento">
                      <label for="nacimiento">fecha de nacim.</label>
                    </div>
                  </td>
                  <td>
                    <button class="btn waves-effect waves-light center" type="submit" name="ac\
eptar">
                      Añadir
                      <i class="material-icons">check_circle</i>
                    </button>
                  </td>
                </tr>
              </form>
            </table>
          </div>
      </div>
    </div>

    <%
      conexion.close();
    %>
    <script type="text/javascript" src="https://code.jquery.com/jquery-2.1.1.min.js"></script>
    <!-- Materialize -->
    <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/js/materialize.min.\
js"></script>

    <script>
      $('.datepicker').pickadate({
        // The title label to use for the month nav buttons
        labelMonthNext: 'Mes siguiente',
        labelMonthPrev: 'Mes anterior',

        // The title label to use for the dropdown selectors
        labelMonthSelect: 'Selecciona un mes',
        labelYearSelect: 'Selecciona un año',
```

```
        // Months and weekdays
        monthsFull: [ 'Enero', 'Febrero', 'Marzo', 'Abril', 'Mayo', 'Junio', 'Julio', 'Agosto'\
, 'Septiembre', 'Octubre', 'Noviembre', 'Diciembre' ],
        monthsShort: [ 'Ene', 'Feb', 'Mar', 'Abr', 'May', 'Jun', 'Jul', 'Ago', 'Sep', 'Oct', '\
Nov', 'Dic' ],
        weekdaysFull: [ 'Domingo', 'Lunes', 'Martes', 'Miércoles', 'Jueves', 'Viernes', 'Sábad\
o' ],
        weekdaysShort: [ 'Dom', 'Lun', 'Mar', 'Mié', 'Jue', 'Vie', 'Sab' ],

        // Materialize modified
        weekdaysLetter: [ 'D', 'L', 'M', 'X', 'J', 'V', 'S' ],

        // Today and clear
        today: 'Hoy',
        clear: 'Limpiar',
        close: 'Cerrar',
        format: "yyyy-mm-dd",
        showOtherMonths: true,
        selectOtherMonths: true,
        selectMonths: true, // Creates a dropdown to control month
        selectYears: 100, // Creates a dropdown of 15 years to control year
        max: true,
        firstDay: 1
      });
    </script>
  </body>
</html>
```

Fichero: modificaCliente.jsp

```
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.Connection"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <!-- Materialize -->
    <link href="http://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/css\
/materialize.min.css">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.4.0/css/font-a\
wesome.min.css">
  </head>
```

```
  <body>
    <% request.setCharacterEncoding("UTF-8"); %>
    <div class="container">
      <div class="row"></div>
      <div class="row">
        <div class="row col m3"></div>
        <div class="col m6 card-panel grey lighten-5">
          <h5 class="center">Datos del cliente</h5>
          <form method="post" action="grabaCliente.jsp">
            <div class="input-field blue-text">
              <i class="material-icons prefix">label_outline</i>
              <input type="number" name="clienteid" id="clienteid" value="<%=request.getParame\
ter("clienteid") %>" readonly>
              <label for="clienteid">código</label>
            </div>
            <div class="input-field">
              <i class="material-icons prefix">perm_identity</i>
              <input type="text" name="nombre" id="nombre" value="<%=request.getParameter("nom\
bre") %>" required>
              <label for="nombre">nombre</label>
            </div>
            <div class="input-field">
              <i class="material-icons prefix">location_on</i>
              <input type="text" name="direccion" id="direccion" value="<%=request.getParamete\
r("direccion") %>" required>
              <label for="direccion">dirección</label>
            </div>
            <div class="input-field">
              <i class="material-icons prefix">call</i>
              <input type="text" name="telefono" id="telefono" value="<%=request.getParameter(\
"telefono") %>" required>
              <label for="telefono">teléfono</label>
            </div>
            <div class="input-field">
              <i class="fa fa-birthday-cake prefix" style="color: #009688"></i>
              <input type="date" class="datepicker" name="nacimiento" id="nacimiento" value="<\
%=request.getParameter("nacimiento") %>" required>
              <label for="nacimiento">fecha de nacim.</label>
            </div>
            <p class="center">
              <button class="btn waves-effect waves-light center" type="submit" name="aceptar">
              Aceptar
              <i class="material-icons">check_circle</i>
              </button>
              <a href="index.jsp" class="btn waves-effect waves-light center red">
              Cancelar
```

```
              <i class="fa fa-times"></i>
              </a>
            </p>
            <br>
          </form>
        </div>
      </div>
    </div>

    <!--Import jQuery before materialize.js-->
    <script type="text/javascript" src="https://code.jquery.com/jquery-2.1.1.min.js"></script>
    <!-- Materialize -->
    <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/js/materialize.min.\
js"></script>

    <script>
      $('.datepicker').pickadate({
        // The title label to use for the month nav buttons
        labelMonthNext: 'Mes siguiente',
        labelMonthPrev: 'Mes anterior',

        // The title label to use for the dropdown selectors
        labelMonthSelect: 'Selecciona un mes',
        labelYearSelect: 'Selecciona un año',

        // Months and weekdays
        monthsFull: [ 'Enero', 'Febrero', 'Marzo', 'Abril', 'Mayo', 'Junio', 'Julio', 'Agosto'\
, 'Septiembre', 'Octubre', 'Noviembre', 'Diciembre' ],
        monthsShort: [ 'Ene', 'Feb', 'Mar', 'Abr', 'May', 'Jun', 'Jul', 'Ago', 'Sep', 'Oct', '\
Nov', 'Dic' ],
        weekdaysFull: [ 'Domingo', 'Lunes', 'Martes', 'Miércoles', 'Jueves', 'Viernes', 'Sábad\
o' ],
        weekdaysShort: [ 'Dom', 'Lun', 'Mar', 'Mié', 'Jue', 'Vie', 'Sab' ],

        // Materialize modified
        weekdaysLetter: [ 'D', 'L', 'M', 'X', 'J', 'V', 'S' ],

        // Today and clear
        today: 'Hoy',
        clear: 'Limpiar',
        close: 'Cerrar',
        format: "yyyy-mm-dd",
        showOtherMonths: true,
        selectOtherMonths: true,
        selectMonths: true, // Creates a dropdown to control month
        selectYears: 100, // Creates a dropdown of 15 years to control year
```

```
        max: true,
        firstDay: 1
      });
    </script>
  </body>
</html>
```

Fichero: altaCliente.jsp

```
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.Connection"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <!-- Materialize -->
    <link href="http://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/css\
/materialize.min.css">
  </head>
  <body>
    <%
      Class.forName("com.mysql.jdbc.Driver");
      Connection conexion = DriverManager.getConnection("jdbc:mysql://localhost:3306/banco", "\
root", "root");
      Statement s = conexion.createStatement();

      request.setCharacterEncoding("UTF-8");

      // Comprueba si el clienteid de cliente ya existe
      String consulta = "SELECT COUNT(*) FROM cliente WHERE clienteid='"
              + request.getParameter("clienteid")
              + "'";

      ResultSet coincidencias = s.executeQuery(consulta);
      coincidencias.next();

      if (coincidencias.getInt(1) != 0) {
        out.print("<script type=\"text/javascript\">alert(\"Lo siento, el código " + request.g\
etParameter("clienteid") + " ya existe\");</script>");
        out.print("<script>document.location = \"index.jsp\"</script>");
      } else {
        // Inserta los datos en la base de datos
```

```
        String insercion = "INSERT INTO cliente VALUES ('"
                + request.getParameter("clienteid")
                + "', '" + request.getParameter("nombre")
                + "', '" + request.getParameter("direccion")
                + "', '" + request.getParameter("telefono")
                + "', '" + request.getParameter("nacimiento") + "')";
        out.print(insercion);
        s.execute(insercion);
        conexion.close();
      }
    %>
    <script>document.location = "index.jsp"</script>
  </body>
</html>
```

Ejercicio 5

Fichero: buscaCliente.jsp

```
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.Connection"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <!-- Materialize -->
    <link href="http://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/css\
/materialize.min.css">

    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.4.0/css/font-a\
wesome.min.css">

    <title>Gestibank</title>
  </head>
  <body>
    <%
      Class.forName("com.mysql.jdbc.Driver");
      Connection conexion = DriverManager.getConnection("jdbc:mysql://localhost:3306/banco","r\
oot", "root");
      Statement s = conexion.createStatement();
```

```
      request.setCharacterEncoding("UTF-8");

      ResultSet listado = s.executeQuery ("SELECT COUNT(*) FROM cliente WHERE nombre LIKE '%" \
+ request.getParameter("nombre") + "%'");
      listado.next();
      int coincidencias = listado.getInt(1);

      // Si no se encuentra ningún cliente con el nombre especificado,
      // volvemos a la página principal.
      if (coincidencias == 0) {
        out.println("<script>document.location = \"index.jsp\"</script>");
      }

      ResultSet listado2 = s.executeQuery ("SELECT * FROM cliente WHERE nombre LIKE '%" + requ\
est.getParameter("nombre") + "%'");

      // Si hay un único cliente con el nombre especificado, vamos directamente
      // a la página de detalle.
      if (coincidencias == 1) {
        listado2.next();
        out.println("<script>document.location = \"detalleCliente.jsp?clienteid=" + listado2.g\
etInt("clienteid") + "\"</script>");
      }

      // Si hay varios clientes cuyos nombres coinciden con el patrón buscado,
      // se muestran todos esos nombres para que el usuario elija.
      if (coincidencias > 1) {
        %>
          <div class="container">
            <div class="row"></div>
            <div class="row">
              <div class="col s3 m3 l3"> </div>
              <div class="col s6 m6 l6">
                <div class="card grey lighten-5">
                  <h3 class="center">G e s t i b a n k</h3>
                  <table class="bordered centered responsive-table">
                  <thead><th>Nombre</th><th></th></thead>
                  <%
                  while (listado2.next()) {
                    %>
                    <tr>
                      <td><%=listado2.getString("nombre") %></td>
                      <form method="post" action="detalleCliente.jsp">
                        <input type="hidden" name="clienteid" value="<%=listado2.getString("cl\
ienteid") %>">

                        <td>
```

```
                        <button class="btn waves-effect waves-light center purple lighten-2" t\
ype="submit" name="editar">
                          <i class="fa fa-plus-circle"></i> Detalle
                        </button>
                        </td>
                      </form>
                    </tr>
                    <%
                  } // while
                  %>
                </table>
              </div>
            </div>
            <div class="col s3 m3 l3"></div>
      <%
      } // if

      conexion.close();
    %>
    <script type="text/javascript" src="https://code.jquery.com/jquery-2.1.1.min.js"></script>
    <!-- Materialize -->
    <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/js/materialize.min.\
js"></script>
  </body>
</html>
```

Fichero: grabaCliente.jsp

```
<%@page import="java.util.Vector"%>
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.Connection"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  </head>
  <body>
    <%
      Class.forName("com.mysql.jdbc.Driver");
      Connection conexion = DriverManager.getConnection("jdbc:mysql://localhost:3306/banco","r\
oot", "root");
      Statement s = conexion.createStatement();
```

```
      request.setCharacterEncoding("UTF-8");

      s.execute("UPDATE cliente SET nombre='" + request.getParameter("nombre") + "', direccion\
='" + request.getParameter("direccion") + "', telefono='" + request.getParameter("telefono") +\
 "', nacimiento='" + request.getParameter("nacimiento") + "' WHERE clienteid=" + request.getPa\
rameter("clienteid"));

      conexion.close();
    %>
    <script>document.location = "index.jsp"</script>
  </body>
</html>
```

Fichero: borraCliente.jsp

```
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.Connection"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    </head>
    <body>
      <%
            Class.forName("com.mysql.jdbc.Driver");
            Connection conexion = DriverManager.getConnection("jdbc:mysql://localhost:3306/ban\
co","root", "root");
            Statement s = conexion.createStatement();

            s.execute ("DELETE FROM cliente WHERE clienteid=" + request.getParameter("clientei\
d"));
        %>
       <script>document.location = "index.jsp"</script>
    </body>
</html>
```

Fichero: index.jsp

```
<%--
  5. Amplía el programa anterior para que se pueda hacer una búsqueda por
     nombre. El programa buscará la cadena introducida dentro del campo "nombre"
     y, si hay varias coincidencias, se mostrará una lista de clientes para
     poder seleccionar uno de ellos y ver todos los datos. Si solo hay una
     coincidencia, se mostrarán directamente los datos del cliente en cuestión.
--%>

<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.Connection"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <!-- Materialize -->
    <link href="http://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/css\
/materialize.min.css">

    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.4.0/css/font-a\
wesome.min.css">

    <title>Gestibank</title>
  </head>

  <body>
    <%
      Class.forName("com.mysql.jdbc.Driver");
      Connection conexion = DriverManager.getConnection("jdbc:mysql://localhost:3306/banco", "\
root", "root");
      Statement s = conexion.createStatement();

      request.setCharacterEncoding("UTF-8");

      ResultSet listado = s.executeQuery("SELECT * FROM cliente");
    %>

    <div class="container">
      <div class="row"></div>
      <div class="row s12 m12 l12">
        <div class="card grey lighten-5">
          <h3 class="center">G e s t i b a n k</h3>
          <table class="bordered centered responsive-table">
```

```
            <thead><th>Código</th><th>Nombre</th><th>Dirección</th><th>Teléfono</th><th>Fecha \
de nacimiento</th><th></th><th></th></thead>
              <%
                while (listado.next()) {
              %>
            <tr>
              <td><%=listado.getString("clienteid")%></td>
              <td><%=listado.getString("nombre")%></td>
              <td><%=listado.getString("direccion")%></td>
              <td><%=listado.getString("telefono")%></td>
              <td><%=listado.getString("nacimiento")%></td>
            <form method="post" action="modificaCliente.jsp">
              <input type="hidden" name="clienteid" value="<%=listado.getString("clienteid")%>\
">
              <input type="hidden" name="nombre" value="<%=listado.getString("nombre")%>">
              <input type="hidden" name="direccion" value="<%=listado.getString("direccion")%>\
">
              <input type="hidden" name="telefono" value="<%=listado.getString("telefono")%>">
              <input type="hidden" name="nacimiento" value="<%=listado.getString("nacimiento")\
%>">
              <td>
                <button class="btn waves-effect waves-light center blue" type="submit" name="e\
ditar">
                  Editar
                </button>
              </td>
            </form>
            <form method="post" action="borraCliente.jsp">
              <input type="hidden" name="clienteid" value="<%=listado.getString("clienteid")%>\
">
              <td>
                <button class="btn waves-effect waves-light center red" type="submit" name="bo\
rrar">
                  Borrar
                </button>
              </td>
            </form>
            </tr>
            <%
              }
            %>
            <form method="post" action="altaCliente.jsp">
              <tr>
                <td>
                  <div class="input-field">
                    <input type="number" name="clienteid" id="clienteid" required>
```

```
              <label for="clienteid">código</label>
            </div>
          </td>
          <td>
            <div class="input-field">
              <input type="text" name="nombre" id="nombre">
              <label for="nombre">nombre</label>
            </div>
          </td>
          <td>
            <div class="input-field">
              <input type="text" name="direccion" id="direccion">
              <label for="direccion">dirección</label>
            </div>
          </td>
          <td>
            <div class="input-field">
              <input type="tel" name="telefono" id="telefono">
              <label for="telefono">teléfono</label>
            </div>
          </td>
          <td>
            <div class="input-field">
              <input type="date" class="datepicker" name="nacimiento" id="nacimiento">
              <label for="nacimiento">fecha de nacim.</label>
            </div>
          </td>
          <td>
            <button class="btn waves-effect waves-light center" type="submit" name="acep\
tar">
              Añadir
            </button>
          </td>
        </tr>
      </form>
      <form method="post" action="buscaCliente.jsp">
        <tr>
          <td>
            <div class="input-field">
              <input type="text" name="nombre" id="buscaNombre">
              <label for="buscaNombre">nombre</label>
            </div>
          </td>
          <td>
            <button class="btn waves-effect waves-light center" type="submit" name="acep\
tar">
```

```
                    <i class="fa fa-search"></i> Buscar
                  </button>
                </td>
              </tr>
            </form>
          </table>
        </div>
      </div>
    </div>

    <%
      conexion.close();
    %>
    <script type="text/javascript" src="https://code.jquery.com/jquery-2.1.1.min.js"></script>
    <!-- Materialize -->
    <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/js/materialize.min.\
js"></script>

    <script>
      $('.datepicker').pickadate({
        // The title label to use for the month nav buttons
        labelMonthNext: 'Mes siguiente',
        labelMonthPrev: 'Mes anterior',
        // The title label to use for the dropdown selectors
        labelMonthSelect: 'Selecciona un mes',
        labelYearSelect: 'Selecciona un año',
        // Months and weekdays
        monthsFull: ['Enero', 'Febrero', 'Marzo', 'Abril', 'Mayo', 'Junio', 'Julio', 'Agosto',\
 'Septiembre', 'Octubre', 'Noviembre', 'Diciembre'],
        monthsShort: ['Ene', 'Feb', 'Mar', 'Abr', 'May', 'Jun', 'Jul', 'Ago', 'Sep', 'Oct', 'N\
ov', 'Dic'],
        weekdaysFull: ['Domingo', 'Lunes', 'Martes', 'Miércoles', 'Jueves', 'Viernes', 'Sábado\
'],
        weekdaysShort: ['Dom', 'Lun', 'Mar', 'Mié', 'Jue', 'Vie', 'Sab'],
        // Materialize modified
        weekdaysLetter: ['D', 'L', 'M', 'X', 'J', 'V', 'S'],
        // Today and clear
        today: 'Hoy',
        clear: 'Limpiar',
        close: 'Cerrar',
        format: "yyyy-mm-dd",
        showOtherMonths: true,
        selectOtherMonths: true,
        selectMonths: true, // Creates a dropdown to control month
        selectYears: 100, // Creates a dropdown of 15 years to control year
        max: true,
```

```
        firstDay: 1
      });
    </script>
  </body>
</html>
```

Fichero: detalleCliente.jsp

```
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.Connection"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <!-- Materialize -->
    <link href="http://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/css\
/materialize.min.css">

    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.4.0/css/font-a\
wesome.min.css">

    <title>Gestibank</title>
  </head>
  <body>
    <%
      Class.forName("com.mysql.jdbc.Driver");
      Connection conexion = DriverManager.getConnection("jdbc:mysql://localhost:3306/banco","r\
oot", "root");
      Statement s = conexion.createStatement();

      request.setCharacterEncoding("UTF-8");

      ResultSet listado = s.executeQuery ("SELECT * FROM cliente WHERE clienteid='" + request.\
getParameter("clienteid") +"'");

      listado.next();
    %>

    <div class="container">
      <div class="row"></div>
      <div class="row">
        <div class="row col m3"></div>
```

```
        <div class="col m6 card-panel grey lighten-5">
          <h5 class="center"><%=listado.getString("nombre") %></h5>
            <p>
              <i class="material-icons teal-text">label_outline</i> Código:
              <b><%=listado.getString("clienteid") %></b>
            </p>
            <p>
              <i class="material-icons teal-text">perm_identity</i> Nombre:
              <b><%=listado.getString("nombre") %></b>
            </p>
            <p>
              <i class="material-icons teal-text">location_on</i> Dirección:
              <b><%=listado.getString("direccion") %></b>
            </p>
            <p>
              <i class="material-icons teal-text">call</i> Teléfono:
              <b><%=listado.getString("telefono") %></b>
            </p>
            <p>
              <i class="fa fa-birthday-cake teal-text" style="color: #009688"> Fecha de nacimi\
ento: </i>
              <b><%=listado.getString("nacimiento") %></b>
            </p>
            <br>
            <p class="center">
              <a href="index.jsp" class="btn waves-effect waves-light center">
                Aceptar
                <i class="material-icons">check_circle</i>
              </a>
            </p>
            <br>
          </form>
        </div>
      </div>
    </div>
    <% conexion.close(); %>

    <script type="text/javascript" src="https://code.jquery.com/jquery-2.1.1.min.js"></script>
    <!-- Materialize -->
    <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/js/materialize.min.\
js"></script>

  </body>
</html>
```

Fichero: modificaCliente.jsp

```
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.Connection"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <!-- Materialize -->
    <link href="http://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/css\
/materialize.min.css">

    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.4.0/css/font-a\
wesome.min.css">

  </head>
  <body>
    <% request.setCharacterEncoding("UTF-8"); %>
    <div class="container">
      <div class="row"></div>
      <div class="row">
        <div class="row col m3"></div>
        <div class="col m6 card-panel grey lighten-5">
          <h5 class="center">Datos del cliente</h5>
          <form method="post" action="grabaCliente.jsp">
            <div class="input-field blue-text">
              <i class="material-icons prefix">label_outline</i>
              <input type="number" name="clienteid" id="clienteid" value="<%=request.getParame\
ter("clienteid") %>" readonly>
              <label for="clienteid">código</label>
            </div>
            <div class="input-field">
              <i class="material-icons prefix">perm_identity</i>
              <input type="text" name="nombre" id="nombre" value="<%=request.getParameter("nom\
bre") %>" required>
              <label for="nombre">nombre</label>
            </div>
            <div class="input-field">
              <i class="material-icons prefix">location_on</i>
              <input type="text" name="direccion" id="direccion" value="<%=request.getParamete\
r("direccion") %>" required>
              <label for="direccion">dirección</label>
            </div>
            <div class="input-field">
```

```
              <i class="material-icons prefix">call</i>
              <input type="text" name="telefono" id="telefono" value="<%=request.getParameter(\
"telefono") %>" required>
              <label for="telefono">teléfono</label>
            </div>
            <div class="input-field">
              <i class="fa fa-birthday-cake prefix" style="color: #009688"></i>
              <input type="date" class="datepicker" name="nacimiento" id="nacimiento" value="<\
%=request.getParameter("nacimiento") %>" required>
              <label for="nacimiento">fecha de nacim.</label>
            </div>
            <p class="center">
              <button class="btn waves-effect waves-light center" type="submit" name="aceptar">
              Aceptar
              <i class="mdi-action-check-circle"></i>
              </button>
              <a href="index.jsp" class="btn waves-effect waves-light center red">
              Cancelar
              <i class="fa fa-times"></i>
              </a>
            </p>
            <br>
          </form>
        </div>
      </div>
    </div>

    <!--Import jQuery before materialize.js-->
    <script type="text/javascript" src="https://code.jquery.com/jquery-2.1.1.min.js"></script>
    <!-- Materialize -->
    <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/js/materialize.min.\
js"></script>

    <script>
      $('.datepicker').pickadate({
        // The title label to use for the month nav buttons
        labelMonthNext: 'Mes siguiente',
        labelMonthPrev: 'Mes anterior',

        // The title label to use for the dropdown selectors
        labelMonthSelect: 'Selecciona un mes',
        labelYearSelect: 'Selecciona un año',

        // Months and weekdays
        monthsFull: [ 'Enero', 'Febrero', 'Marzo', 'Abril', 'Mayo', 'Junio', 'Julio', 'Agosto'\
, 'Septiembre', 'Octubre', 'Noviembre', 'Diciembre' ],
```

```
        monthsShort: [ 'Ene', 'Feb', 'Mar', 'Abr', 'May', 'Jun', 'Jul', 'Ago', 'Sep', 'Oct', '\
Nov', 'Dic' ],
        weekdaysFull: [ 'Domingo', 'Lunes', 'Martes', 'Miércoles', 'Jueves', 'Viernes', 'Sábad\
o' ],
        weekdaysShort: [ 'Dom', 'Lun', 'Mar', 'Mié', 'Jue', 'Vie', 'Sab' ],

        // Materialize modified
        weekdaysLetter: [ 'D', 'L', 'M', 'X', 'J', 'V', 'S' ],

        // Today and clear
        today: 'Hoy',
        clear: 'Limpiar',
        close: 'Cerrar',
        format: "yyyy-mm-dd",
        showOtherMonths: true,
        selectOtherMonths: true,
        selectMonths: true, // Creates a dropdown to control month
        selectYears: 100, // Creates a dropdown of 15 years to control year
        max: true,
        firstDay: 1
      });
    </script>
  </body>
</html>
```

Fichero: altaCliente.jsp

```
<%--
    Document   : alta_cliente
    Created on : 08-may-2013, 12:51:46
    Author     : luisjose
--%>

<%@page import="java.sql.Statement"%>
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.Connection"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <!-- Materialize -->
    <link href="http://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/css\
/materialize.min.css">
```

```
  </head>
  <body>
    <%
      Class.forName("com.mysql.jdbc.Driver");
      Connection conexion = DriverManager.getConnection("jdbc:mysql://localhost:3306/banco", "\
root", "root");
      Statement s = conexion.createStatement();

      request.setCharacterEncoding("UTF-8");

      // Comprueba si el clienteid de cliente ya existe
      String consulta = "SELECT COUNT(*) FROM cliente WHERE clienteid='"
              + request.getParameter("clienteid")
              + "'";

      ResultSet coincidencias = s.executeQuery(consulta);
      coincidencias.next();

      if (coincidencias.getInt(1) != 0) {
        out.print("<script type=\"text/javascript\">alert(\"Lo siento, el código " + request.g\
etParameter("clienteid") + " ya existe\");</script>");
        out.print("<script>document.location = \"index.jsp\"</script>");
      } else {
        // Inserta los datos en la base de datos
        String insercion = "INSERT INTO cliente VALUES ('"
                + request.getParameter("clienteid")
                + "', '" + request.getParameter("nombre")
                + "', '" + request.getParameter("direccion")
                + "', '" + request.getParameter("telefono")
                + "', '" + request.getParameter("nacimiento") + "')";
        out.print(insercion);
        s.execute(insercion);
        conexion.close();
      }
    %>
    <script>document.location = "index.jsp"</script>
  </body>
</html>
```

Ejercicio 6

Fichero: modificaStock.jsp

```
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.Connection"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <link href="http://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/css\
/materialize.min.css">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.4.0/css/font-a\
wesome.min.css">
  </head>
  <body>
    <% request.setCharacterEncoding("UTF-8"); %>
    <div class="container">
      <div class="row"></div>
      <div class="row">
        <div class="row col m3"></div>
        <div class="col m6 card-panel grey lighten-5">
          <h5 class="center">Modificación del stock</h5>
          <form method="post" action="grabaArticulo.jsp">
            <div class="input-field blue-text">
              <i class="fa fa-tag prefix"></i>
              <input type="number" name="codigo" id="clienteid" value="<%=request.getParameter\
("codigo") %>" readonly>
              <label for="codigo">código</label>
            </div>
            <div class="input-field blue-text">
              <i class="fa fa-file-text-o prefix"></i>
              <input type="text" name="descripcion" id="descripcion" value="<%=request.getPara\
meter("descripcion") %>" readonly>
              <label for="descripcion">descripción</label>
            </div>
            <div class="input-field blue-text">
              <i class="fa fa-eur prefix"></i>
              <input type="number" min="0" step="0.01" name="preciocompra" id="preciocompra" v\
alue="<%=request.getParameter("preciocompra") %>" readonly>
              <label for="preciocompra">precio de compra</label>
            </div>
            <div class="input-field blue-text">
              <i class="fa fa-eur prefix"></i>
              <input type="number" min="0" step="0.01" name="precioventa" id="precioventa" val\
ue="<%=request.getParameter("precioventa") %>" readonly>
```

```
          <label for="precioventa">precio de venta</label>
        </div>
        <div class="input-field">
          <i class="fa fa-database prefix"></i>
          <input type="number" min="0" name="stock" id="stock" value="<%=request.getParame\
ter("stock") %>" required>
          <label for="stock">stock</label>
        </div>
        <p class="center">
          <button class="btn waves-effect waves-light center" type="submit" name="aceptar">
          Aceptar
          <i class="material-icons">check_circle</i>
          </button>
          <a href="index.jsp" class="btn waves-effect waves-light center red">
          Cancelar
          <i class="fa fa-times"></i>
          </a>
        </p>
        <br>
      </form>
    </div>
  </div>
</div>

<script type="text/javascript" src="https://code.jquery.com/jquery-2.1.1.min.js"></script>
<!-- Materialize -->
<script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/js/materialize.min.\
js"></script>

  </body>
</html>
```

Fichero: altaArticulo.jsp

```
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.Connection"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  </head>
  <body>
    <%
```

```
      Class.forName("com.mysql.jdbc.Driver");
      Connection conexion = DriverManager.getConnection("jdbc:mysql://localhost:3306/gestisima\
l","root", "root");
      Statement s = conexion.createStatement();

      request.setCharacterEncoding("UTF-8");

      // Comprueba si el codigo de articulo ya existe
      String consulta = "SELECT COUNT(*) FROM articulo WHERE codigo='"
                + request.getParameter("codigo")
                + "'";

      ResultSet coincidencias = s.executeQuery(consulta);
      coincidencias.next();

      if (coincidencias.getInt(1) != 0) {
        out.print("<script type=\"text/javascript\">alert(\"Lo siento, el código " + request.g\
etParameter("codigo") + " ya existe\");</script>");
        out.print("<script>document.location = \"index.jsp\"</script>");
      } else {
        // Inserta los datos en la base de datos
        String insercion = "INSERT INTO articulo VALUES ('"
                   + request.getParameter("codigo")
                   + "', '" + request.getParameter("descripcion")
                   + "', " + request.getParameter("preciocompra")
                   + ", " + request.getParameter("precioventa")
                   + ", " + request.getParameter("stock")+ ")";
        out.print(insercion);
        s.execute(insercion);
        conexion.close();
      }
    %>
    <script>document.location = "index.jsp"</script>
  </body>
</html>
```

Fichero: index.jsp

```
<%--
  6. Crea una versión web del programa GESTISIMAL (GESTIón SIMplifcada de
     Almacén) para llevar el control de los artículos de un almacén. De cada
     artículo se debe saber el código, la descripción, el precio de compra, el
     precio de venta y el stock (número de unidades). La entrada y salida de
     mercancía supone respectivamente el incremento y decremento de stock de un
     determinado artículo. Hay que controlar que no se pueda sacar más mercancía
     de la que hay en el almacén. Aprovecha las opciones que puede ofrecer una
     interfaz web, por ejemplo para eliminar un artículo ya no será necesario
     pedir el código sino que se puede mostrar un listado de todos los artículos
     de tal forma que se puedan borrar un artículo directamente pulsando un
     botón.
--%>
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.Connection"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <link href="http://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/css\
/materialize.min.css">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.4.0/css/font-a\
wesome.min.css">
    <title>Gestisimal</title>
  </head>
  <body>
    <%
      Class.forName("com.mysql.jdbc.Driver");
      Connection conexion = DriverManager.getConnection("jdbc:mysql://localhost:3306/gestisima\
l","root", "root");
      Statement s = conexion.createStatement();

      request.setCharacterEncoding("UTF-8");

      ResultSet listado = s.executeQuery ("SELECT * FROM articulo");
    %>


    <div class="container">
      <div class="row"></div>
      <div class="row s12 m12 l12">
        <div class="card grey lighten-5">
```

```
          <h3 class="center">G e s t i s i m a l</h3>
          <table class="bordered centered responsive-table">
            <thead><th>Código</th><th>Descripción</th><th>Precio de compra</th><th>Precio de v\
enta</th><th>Stock</th><th></th><th></th><th></th></thead>
            <%
              while (listado.next()) {
            %>
                <tr>
                  <td><%=listado.getString("codigo") %></td>
                  <td><%=listado.getString("descripcion") %></td>
                  <td><%=listado.getString("precio_compra") %></td>
                  <td><%=listado.getString("precio_venta") %></td>
                  <td><%=listado.getString("stock") %></td>
                  <form method="post" action="modificaArticulo.jsp">
                    <input type="hidden" name="codigo" value="<%=listado.getString("codigo") %\
>">
                    <input type="hidden" name="descripcion" value="<%=listado.getString("descr\
ipcion") %>">
                    <input type="hidden" name="preciocompra" value="<%=listado.getString("prec\
io_compra") %>">
                    <input type="hidden" name="precioventa" value="<%=listado.getString("preci\
o_venta") %>">
                    <input type="hidden" name="stock" value="<%=listado.getString("stock") %>">
                    <td>
                    <button class="btn waves-effect waves-light center blue" type="submit" nam\
e="editar" title="Modificar artículo">
                      <i class="fa fa-pencil"></i>
                    </button>
                    </td>
                  </form>
                  <form method="post" action="borraArticulo.jsp">
                    <input type="hidden" name="codigo" value="<%=listado.getString("codigo") %\
>">
                    <td>
                    <button class="btn waves-effect waves-light center red" type="submit" name\
="borrar" title="Borrar artículo">
                      <i class="fa fa-times"></i>
                    </button>
                    </td>
                  </form>
                  <form method="post" action="modificaStock.jsp">
                    <input type="hidden" name="codigo" value="<%=listado.getString("codigo") %\
>">
                    <input type="hidden" name="descripcion" value="<%=listado.getString("descr\
ipcion") %>">
                    <input type="hidden" name="preciocompra" value="<%=listado.getString("prec\
```

```
io_compra") %>">
                    <input type="hidden" name="precioventa" value="<%=listado.getString("preci\
o_venta") %>">
                    <input type="hidden" name="stock" value="<%=listado.getString("stock") %>">
                    <td>
                    <button class="btn waves-effect waves-light center green" type="submit" na\
me="editar" title="Entrada/Salida de stock">
                      <i class="fa fa-database"></i>
                    </button>
                    </td>
                  </form>
                </tr>
            <%
              }
            %>
            <form method="post" action="altaArticulo.jsp">
              <tr>
                <td>
                  <div class="input-field">
                    <input type="number" name="codigo" id="codigo" required>
                    <label for="codigo">código</label>
                  </div>
                </td>
                <td>
                  <div class="input-field">
                    <input type="text" name="descripcion" id="descripcion">
                    <label for="descripcion">descripción</label>
                  </div>
                </td>
                <td>
                  <div class="input-field">
                    <input type="number" min="0" step="0.01" name="preciocompra" id="preciocom\
pra">

                    <label for="preciocompra">precio de compra</label>
                  </div>
                </td>
                <td>
                  <div class="input-field">
                    <input type="number" min="0" step="0.01" name="precioventa" id="preciovent\
a">

                    <label for="precioventa">precio de venta</label>
                  </div>
                </td>
                <td>
                  <div class="input-field">
                    <input type="number" min="0" name="stock" id="stock">
```

```
                  <label for="stock">stock</label>
                </div>
              </td>
              <td>
                <button class="btn waves-effect waves-light center" type="submit" name="acep\
tar" title="Añadir artículo">
                  <i class="fa fa-plus"></i>
                </button>
              </td>
            </tr>
          </form>
        </table>
      </div>
    </div>
  </div>

  <%
    conexion.close();
  %>
  <script type="text/javascript" src="https://code.jquery.com/jquery-2.1.1.min.js"></script>
  <!-- Materialize -->
  <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/js/materialize.min.\
js"></script>

  </body>
</html>
```

Fichero: modificaArticulo.jsp

```
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.Connection"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <link href="http://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/css\
/materialize.min.css">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.4.0/css/font-a\
wesome.min.css">
  </head>
  <body>
    <% request.setCharacterEncoding("UTF-8"); %>
```

```
    <div class="container">
      <div class="row"></div>
      <div class="row">
        <div class="row col m3"></div>
        <div class="col m6 card-panel grey lighten-5">
          <h5 class="center">Datos del artículo</h5>
          <form method="post" action="grabaArticulo.jsp">
            <div class="input-field blue-text">
              <i class="fa fa-tag prefix"></i>
              <input type="number" name="codigo" id="clienteid" value="<%=request.getParameter\
("codigo") %>" readonly>
              <label for="codigo">código</label>
            </div>
            <div class="input-field">
              <i class="fa fa-file-text-o prefix"></i>
              <input type="text" name="descripcion" id="descripcion" value="<%=request.getPara\
meter("descripcion") %>" required>
              <label for="descripcion">descripción</label>
            </div>
            <div class="input-field">
              <i class="fa fa-eur prefix"></i>
              <input type="number" min="0" step="0.01" name="preciocompra" id="preciocompra" v\
alue="<%=request.getParameter("preciocompra") %>" required>
              <label for="preciocompra">precio de compra</label>
            </div>
            <div class="input-field">
              <i class="fa fa-eur prefix"></i>
              <input type="number" min="0" step="0.01" name="precioventa" id="precioventa" val\
ue="<%=request.getParameter("precioventa") %>" required>
              <label for="precioventa">precio de venta</label>
            </div>
            <div class="input-field">
              <i class="fa fa-database prefix"></i>
              <input type="number" min="0" name="stock" id="stock" value="<%=request.getParame\
ter("stock") %>" required>
              <label for="stock">stock</label>
            </div>
            <p class="center">
              <button class="btn waves-effect waves-light center" type="submit" name="aceptar">
              Aceptar
              <i class="material-icons">check_circle</i>
              </button>
              <a href="index.jsp" class="btn waves-effect waves-light center red">
              Cancelar
              <i class="fa fa-times"></i>
              </a>
```

```
          </p>
          <br>
        </form>
      </div>
    </div>
  </div>

  <script type="text/javascript" src="https://code.jquery.com/jquery-2.1.1.min.js"></script>
  <!-- Materialize -->
  <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.6/js/materialize.min.\
js"></script>

  </body>
</html>
```

Fichero: grabaArticulo.jsp

```
<%@page import="java.util.Vector"%>
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.Connection"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Gestisimal</title>
  </head>
  <body>
    <div class="principal">
      <%
        Class.forName("com.mysql.jdbc.Driver");
        Connection conexion = DriverManager.getConnection("jdbc:mysql://localhost:3306/gestisi\
mal", "root", "root");
        Statement s = conexion.createStatement();

        request.setCharacterEncoding("UTF-8");

        s.execute("UPDATE articulo SET descripcion='" + request.getParameter("descripcion") + \
"', precio_compra=" + request.getParameter("preciocompra") + ", precio_venta=" + request.getPa\
rameter("precioventa") + ", stock=" + request.getParameter("stock") + " WHERE codigo=" + reque\
st.getParameter("codigo"));

        conexion.close();
      %>
```

```
      <script>document.location = "index.jsp"</script>
  </body>
</html>
```

Fichero: borraArticulo.jsp

```
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.Connection"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    </head>
    <body>
      <%
            Class.forName("com.mysql.jdbc.Driver");
            Connection conexion = DriverManager.getConnection("jdbc:mysql://localhost:3306/ges\
tisimal","root", "root");
            Statement s = conexion.createStatement();

            s.execute ("DELETE FROM articulo WHERE codigo=" + request.getParameter("codigo"));
        %>
       <script>document.location = "index.jsp"</script>
    </body>
</html>
```

Control de excepciones

Ejercicio 1

Fichero: S14Ejercicio01.java

```java
import java.util.Scanner;

public class S14Ejercicio01 {

  public static void main(String[] args) {
    System.out.println("Por favor, vaya introduciendo números enteros.");

    Scanner s = new Scanner(System.in);

    int maximo = Integer.MIN_VALUE;

    for (int i = 1; i < 7; i++) {

      boolean datoValido = false;
      int numero = 0;

      do {
        try {
          System.out.print("Nº " + i + ": ");
          numero = Integer.parseInt(s.nextLine());
          datoValido = true;
        } catch (Exception e) {
          System.out.println("El dato introducido no es correcto, debe ser un número entero.");
          System.out.println("Por favor, inténtelo de nuevo.");
        }
      } while (!datoValido);

      if (numero > maximo) {
        maximo = numero;
      }
    }

    System.out.println("El valor máximo introducido es " + maximo);
  }

}
```

Ejercicio 2

Fichero: ExcepcionApareamientoImposible.java

```
public class ExcepcionApareamientoImposible extends Exception {
  public ExcepcionApareamientoImposible() {

  }
}
```

Fichero: S14Ejercicio02.java

```
public class S14Ejercicio02 {

  public static void main(String[] args) {
    Gato garfield = new Gato("macho");
    Gato tom = new Gato("macho");
    try {
      garfield.apareaCon(tom);
    } catch (ExcepcionApareamientoImposible eai) {
      eai.printStackTrace();
      System.out.println("Dos gatos del mismo sexo no se pueden aparear.");
    }
  }

}
```

Fichero: Gato.java

```
public class Gato {

  private String color;
  private String raza;
  private String sexo;

  public Gato(String s) {
    this.sexo = s;
  }

  public void maulla() {
    System.out.println("Miauuuu");
  }

  public Gato apareaCon(Gato pareja) throws ExcepcionApareamientoImposible {
    if (this.sexo.equals(pareja.sexo)) {
```

```
      throw new ExcepcionApareamientoImposible();
    }

    String s = (int)(Math.random() * 2) ==  0 ? "hembra" : "macho";
    return new Gato(s);
  }

}
```

Ejercicio 3

Fichero: S14Ejercicio03.java

```
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.ArrayList;

public class S14Ejercicio03 {

  public static void main(String[] args) {
    ArrayList<Exception> excepciones = new ArrayList<Exception>();

    excepciones.add(new NumberFormatException());
    excepciones.add(new IOException());
    excepciones.add(new FileNotFoundException());
    excepciones.add(new IndexOutOfBoundsException());
    excepciones.add(new ArithmeticException());

    try {
      throw excepciones.get((int)(Math.random() * 6));
    } catch (Exception e) {
      e.printStackTrace();
    }
  }

}
```

Sesiones y cookies

Sesiones

Ejercicio 1

Fichero: index.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Contador de visitas</title>
  </head>
  <body>
    <%
      session.setAttribute("n", session.getAttribute("n") == null ?
          1 : (Integer)session.getAttribute("n") + 1);
    %>
    <h1><%= session.getAttribute("n") %></h1>
    <a href="reset.jsp">Reinicia la cuenta</a>
  </body>
</html>
```

Fichero: reset.jsp

```
<%
  session.invalidate();
  response.sendRedirect("index.jsp");
%>
```

Ejercicio 2

Fichero: index.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Capítulo 15 - Ejercicio 2</title>
  </head>
  <body>
    <%
      double n = 0;
      double total = 0;
      int numeros = 0;

      if (session.getAttribute("total") == null) {
        session.setAttribute("total", 0.0);
        session.setAttribute("numeros", 0);
        total = 0;
        numeros = 0;
      } else {
        total = (Double) session.getAttribute("total");
        numeros = (Integer) session.getAttribute("numeros");
        n = Double.parseDouble(request.getParameter("n"));
        session.setAttribute("total", total + n);
        session.setAttribute("numeros", ++numeros);
      }

      if (n >= 0) {
    %>
    <p>Vaya introduciendo números para calcular la media.</p>
    <p>Para terminar, introduzca un número negativo.</p>
    <form action="#" method="post">
      <input type="number" name="n" step="any" autofocus>
      <input type="submit" value="Aceptar">
    </form>
    <%
      } else {
        // Hay que quitar el último número introducido que es negativo
        session.setAttribute("total", (Double) session.getAttribute("total") - n);
        session.setAttribute("numeros", (Integer) session.getAttribute("numeros") - 1);
    %>
    <p>
      La media es
      <%= (Double) session.getAttribute("total") / (Integer) session.getAttribute("numeros")%>
    </p>
    <%
      }
```

```
    %>
  </body>
</html>
```

Ejercicio 3

Fichero: index.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Capítulo 15 - Ejercicio 3</title>
  </head>
  <body>
    <%
      int n = 0;
      int sumaImpares = 0;
      int impares = 0;
      int maximoPar = Integer.MIN_VALUE;

      if (session.getAttribute("sumaImpares") == null) {
        sumaImpares = 0;
        impares = 0;
        maximoPar = Integer.MIN_VALUE;
        session.setAttribute("sumaImpares", sumaImpares);
        session.setAttribute("impares", impares);
        session.setAttribute("maximoPar", maximoPar);
      } else {
        sumaImpares = (Integer) session.getAttribute("sumaImpares");
        impares = (Integer) session.getAttribute("impares");
        maximoPar = (Integer) session.getAttribute("maximoPar");

        n = Integer.parseInt(request.getParameter("n"));

        if (n % 2 == 0) {
          if (n > maximoPar) {
            session.setAttribute("maximoPar", n);
          }
        } else {
          session.setAttribute("sumaImpares", sumaImpares + n);
          session.setAttribute("impares", ++impares);
        }
```

```
      }

      if (n >= 0) {
    %>
    <p>Vaya introduciendo números enteros.</p>
    <p>El programa calculará la media de los impares y el mayor de los pares.</p>
    <p>Para terminar, introduzca un número negativo.</p>
    <form action="#" method="post">
      <input type="number" name="n" autofocus>
      <input type="submit" value="Aceptar">
    </form>
    <%
    } else {
      // Hay que quitar el último número introducido si es impar
      if (n % 2 != 0) {
        session.setAttribute("sumaImpares", (Integer) session.getAttribute("sumaImpares") - n);
        session.setAttribute("impares", (Integer) session.getAttribute("impares") - 1);
      }

    %>
    <p>
      La media de los impares es
      <%= (Integer) session.getAttribute("sumaImpares")  * 1.0 / (Integer) session.getAttribut\
e("impares") %>
      <br>
      El máximo de los pares es <%= (Integer) session.getAttribute("maximoPar")%>
    </p>
    <%
    }
    %>
  </body>
</html>
```

Ejercicio 4

Fichero: index.jsp

```
<%@page import="clases.Carrito"%>
<%@page import="clases.ElementoDeCarrito"%>
<%@page import="java.util.ArrayList"%>
<%@page import="clases.Producto"%>
<%@page import="clases.Catalogo"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Tienda de Estilográficas</title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootst\
rap.min.css" integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1\
T" crossorigin="anonymous">
    <link href="https://fonts.googleapis.com/css?family=Great+Vibes&display=swap" rel="stylesh\
eet">
    <style>
      h1 {
        font-family: 'Great Vibes', cursive;
        font-size: 6em;
      }
      .carrito {
        border: #ffc107 solid 2px;
        border-radius: 6px;
        padding: 4px;
      }
    </style>
  </head>
  <body>
    <%
      if (session.getAttribute("catalogo") == null) {
        Catalogo catalogo = new Catalogo();
        catalogo.cargaDatos();
        session.setAttribute("catalogo", catalogo);
      }

      if (session.getAttribute("carrito") == null) {
        Carrito carrito = new Carrito();
        session.setAttribute("carrito", carrito);
      }
    %>
    <br>
    <h1 class="text-center">Tienda de Estilográficas</h1>

    <div class="container">
      <div class="row">
```

```
        <!-- Catálogo de productos -->

        <div class="col">
          <div class="row">
            <%
              for (Producto p : ((Catalogo) session.getAttribute("catalogo")).getProductos()) {
            %>

            <div class="col-sm-4">
              <div class="card">
                <img src="img/<%= p.getImagen()%>" class="card-img-top img-fluid">
                <div class="card-body">
                  <h4 class="card-title"><%= p.getNombre()%></h4>
                  <h5><%= String.format("%.2f", p.getPrecio()) %> €</h5>
                  <a href="compra.jsp?codigo=<%= p.getCodigo()%>" class="btn btn-primary">
                    Añadir al carrito
                  </a>
                </div>
              </div>
            </div>
            <%
              }
            %>
          </div>
        </div>


        <!-- Catálogo de productos -->

        <div class="col-3">
          <div class="carrito">
            <img src="img/cart.svg" width="36px">
          <%
            for (ElementoDeCarrito e : ((Carrito) session.getAttribute("carrito")).getElemento\
s()) {
          %>
              <div class="card">
                <img src="img/<%= e.getProducto().getImagen()%>" class="card-img-top img-fluid\
">
                <div class="card-body">
                  <%= e.getProducto().getNombre() %><br>
                  <%= String.format("%.2f", e.getProducto().getPrecio()) %> €<br>
                  <%= e.getCantidad() %> unidades<br>
                  <a href="elimina.jsp?codigo=<%= e.getProducto().getCodigo() %>" class="btn b\
tn-warning text-white">
```

```
                    Eliminar
                  </a>
                </div>
              </div>
          <%
            }
          %>
          </div>
        </div>
      </div>
    </div>


    <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965\
DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js" in\
tegrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1" crossorigin=\
"anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js" integ\
rity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM" crossorigin="an\
onymous"></script>
  </body>
</html>
```

Fichero: compra.jsp

```
<%@page import="clases.Carrito"%>
<%
  int codigo = Integer.parseInt(request.getParameter("codigo"));
  Carrito carrito = (Carrito) session.getAttribute("carrito");
  carrito.meteProductoConCodigo(codigo);
  session.setAttribute("carrito", carrito);
  response.sendRedirect("index.jsp");
%>
```

Fichero: elimina.jsp

```
<%@page import="clases.Carrito"%>
<%
  int codigo = Integer.parseInt(request.getParameter("codigo"));
  Carrito carrito = (Carrito) session.getAttribute("carrito");
  carrito.eliminaProductoConCodigo(codigo);
  session.setAttribute("carrito", carrito);
  response.sendRedirect("index.jsp");
%>
```

Cookies

Ejercicio 1

Fichero: index.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Página multi-idioma</title>
    <style>
      a {
        text-decoration: none;
      }
    </style>
  </head>
  <body>
    <%
      String idioma = "es";

      Cookie cookie = dameCookie(request, "idioma");

      if (cookie == null) {
        cookie = new Cookie("idioma", "ES");
        cookie.setPath("/");
        cookie.setMaxAge(365 * 60 * 60);
        response.addCookie(cookie);
      } else {
        idioma = cookie.getValue();
      }
    %>

    <h1>
      <%=idioma.equals("es") ? "Don Quijote de La Mancha"
```

```
          : idioma.equals("eo") ? "Don Quijote de la Mancha"
          : idioma.equals("en") ? "Don Quixote of La Mancha"
          : "Don Quichotte de la Manche"%>
    </h1>

    <object data="<%=idioma.equals("es") ? "txt/es.txt"
          : idioma.equals("eo") ? "txt/eo.txt"
          : idioma.equals("en") ? "txt/en.txt"
          : "fr.txt"%>"></object>

    <br>

    <a href="cambia-idioma.jsp?idioma=es">
      <img src="img/es.svg" width="30" height="15">
    </a>
    <a href="cambia-idioma.jsp?idioma=eo">
      <img src="img/eo.svg" width="30" height="15">
    </a>
    <a href="cambia-idioma.jsp?idioma=en">
      <img src="img/en.svg" width="30" height="15">
    </a>
    <a href="cambia-idioma.jsp?idioma=fr">
      <img src="img/fr.svg" width="30" height="15">
    </a>
  </body>
</html>

<%!
  public static Cookie dameCookie(HttpServletRequest request, String nombre) {
    Cookie[] cookies = request.getCookies();
    if (cookies != null) {
      for (Cookie cookie : cookies) {
        if (cookie.getName().equals(nombre)) {
          return cookie;
        }
      }
    }
    return null;
  }
%>
```

Fichero: cambia-idioma.jsp

```
<%
  Cookie cookie = new Cookie("idioma", request.getParameter("idioma"));
  cookie.setPath("/");
  cookie.setMaxAge(365 * 60 * 60);
  response.addCookie(cookie);

  response.sendRedirect("index.jsp");
%>
```

Fichero: es.txt

```
En un lugar de La Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un \
hidalgo de los de lanza en astillero, adarga antigua, rocín flaco, y galgo corredor.
```

Fichero: eo.txt

```
En la vilaĝo de La Mancha, kies nomon mi ne volas memori, antaŭ nelonge vivis hidalgo el tiuj \
kun lanco en rako, antikva ŝildo, osta ĉevalaĉo kaj rapida lebrelo.
```

Fichero: en.txt

```
Somewhere in La Mancha, in a place whose name I do not care to remember, a gentleman lived not\
 long ago, one of those who has a lance and ancient shield on a shelf and keeps a skinny nag a\
nd a greyhound for racing.
```

Fichero: fr.txt

```
En un village de la Manche, du nom duquel je ne me veux souvenir, demeurait, il n'y a pas long\
temps, un gentilhomme de ceux qui ont lance au râtelier, targe antique, roussin maigre et levr\
ier bon coureur.
```

Ejercicio 2

Fichero: index.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Colores en cookies</title>

    <%
      String texto = "#000000";

      Cookie cookieTexto = dameCookie(request, "texto");

      if (cookieTexto == null) {
        cookieTexto = new Cookie("texto", texto);
        cookieTexto.setPath("/");
        cookieTexto.setMaxAge(365 * 60 * 60);
        response.addCookie(cookieTexto);
      } else {
        texto = cookieTexto.getValue();
      }

      String fondo = "#ffffff";

      Cookie cookieFondo = dameCookie(request, "fondo");

      if (cookieFondo == null) {
        cookieFondo = new Cookie("fondo", fondo);
        cookieFondo.setPath("/");
        cookieFondo.setMaxAge(365 * 60 * 60);
        response.addCookie(cookieFondo);
      } else {
        fondo = cookieFondo.getValue();
      }
    %>

    <style>
      body {
        color: <%= texto %>;
        background-color: <%= fondo %>;
      }
    </style>
  </head>
  <body>
    <h1>Colores en cookies</h1>

    <p>
```

```
      Esta página permite guardar los colores del texto y del fondo.
    </p>

    <form action="cambia-colores.jsp" method="POST">
      Texto <input type="color" name="texto" value="<%= texto %>">
      Fondo <input type="color" name="fondo"  value="<%= fondo %>">
      <input type="submit" value="Aceptar">
    </form>
  </body>
</html>


<%!
  public static Cookie dameCookie(HttpServletRequest request, String nombre) {
    Cookie[] cookies = request.getCookies();
    if (cookies != null) {
      for (Cookie cookie : cookies) {
        if (cookie.getName().equals(nombre)) {
          return cookie;
        }
      }
    }
    return null;
  }
%>
```

Fichero: cambia-colores.jsp

```
<%
  Cookie cookieTexto = new Cookie("texto", request.getParameter("texto"));
  cookieTexto.setPath("/");
  cookieTexto.setMaxAge(365 * 60 * 60);
  response.addCookie(cookieTexto);

  Cookie cookieFondo = new Cookie("fondo", request.getParameter("fondo"));
  cookieFondo.setPath("/");
  cookieFondo.setMaxAge(365 * 60 * 60);
  response.addCookie(cookieFondo);

  response.sendRedirect("index.jsp");
%>
```